[illegible]

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A9973

THE COMPUTER SIMULATION AND MODELING OF A FLEXIBLE MISSILE IN 2-D MOTION

by

Mehmet Aysel

December 1989

Thesis Advisor L.W. Chang

Approved for public release; distribution is unlimited.

nclassified
urity classification of this page

REPORT DOCUMENTATION PAGE

a Report Security Classification Unclassified		1b Restrictive Markings	
a Security Classification Authority		3 Distribution Availability of Report Approved for public release; distribution is unlimited.	
Declassification Downgrading Schedule			
Performing Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
a Name of Performing Organization aval Postgraduate School	6b Office Symbol *(if applicable)* 33	7a Name of Monitoring Organization Naval Postgraduate School	
Address *(city, state, and ZIP code)* Ionterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
a Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	

Program Element No	Project No	Task No	Work Unit Accession No

Title *(include security classification)* THE COMPUTER SIMULATION AND MODELING OF A FLEXIBLE MISSILE IN
-D MOTION

Personal Author(s) Mehmet Aysel

a Type of Report Iaster's Thesis	13b Time Covered From To	14 Date of Report *(year, month, day)* December 1989	15 Page Count 104

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
eld	Group	Subgroup	Flexible Missile, Missile, Control and Dynamics.

Abstract *(continue on reverse if necessary and identify by block number)*

The main goal of this research is to model a flexible missile with structural flexibility utilizing the Equivalent Rigid Link ystem (ERLS) with an enhanced natural mode discretization. Dynamic analysis of the flexible missile in 2-Dimension moon is presented.

Computer simulation is performed where the pitch angle of the missile is controlled with a rigid-body controller. The efcts of increasing payloads and speed to the system performance are discussed.

Distribution/Availability of Abstract unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
a Name of Responsible Individual .W. Chang	22b Telephone *(include Area code)* (408)646-2632	22c Office Symbol 69Ck

) FORM 1473,84 MAR
83 APR edition may be used until exhausted
All other editions are obsolete
security classification of this page

Unclassified

T247831

Approved for public release; distribution is unlimited.

The Computer Simulation and Modeling of a Flexible Missile In 2-D Motion

by

Mehmet Aysel
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1989

ABSTRACT

The main goal of this research is to model a flexible missile with structural flexibility utilizing the Equivalent Rigid Link System (ERLS) with an enhanced natural mode discretization. Dynamic analysis of the flexible missile in 2-Dimension motion is presented.

Computer simulation is performed where the pitch angle of the missile is controlled with a rigid-body controller. The effects of increasing payloads and speed to the system performance are discussed.

C.1

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 1
  A. BACKGROUND 1
  B. LITERATURE REVIEW 1
  C. OUTLINE 3

II. MODEL FORMULATION FOR A FLEXIBLE MISSILE 4
  A. KINEMATICS 4
  B. KINETICS 8

III. THE DEVELOPMENT OF A RIGID-BODY CONTROLLER 12

IV. COMPUTER SIMULATION 16
  A. SIMULATION OBJECTIVES 16
  B. SOLUTION TECHNIQUE 16
  C. THE COMPUTER SIMULATION CODE 17
  D. SIMULATION RESULTS 17

V. SUMMARY 49
  A. CONCLUSIONS 49
  B. RECOMMENDATIONS 49

APPENDIX A. DERIVATION OF THE MODE SHAPE RESPONSE MATRIX FOR THE FLEXIBLE MISSILE 50

APPENDIX B. DERIVATION OF THE EQUATIONS OF MOTION FOR THE FLEXIBLE MISSILE 57

APPENDIX C. DERIVATION OF THE GENERALIZED FORCES OF THE EQUATIONS OF THE MOTION 65

APPENDIX D. COMPUTER SIMULATION CODE 67

LIST OF REFERENCES 91

INITIAL DISTRIBUTION LIST 92

LIST OF FIGURES

Figure 1. The general configuration of the missile 4
Figure 2. Equivalent Rigid Link System (ERLS) 5
Figure 3. Applied Generalized Forces on Flexible Missile 9
Figure 4. The flexible missile with rigid-body controller 12
Figure 5. The desired and actual trajectory for the rigid missile with rigid-body controller 19
Figure 6. The control angle for the rigid missile with rigid-body controller 20
Figure 7. The large motion position X for the rigid missile with rigid-body controller 21
Figure 8. The large motion position Y for the rigid missile with rigid-body controller 22
Figure 9. The desired and actual trajectory for the flexible missile with rigid-body controller 23
Figure 10. The control angle for the flexible missile with rigid-body controller 24
Figure 11. The small motion position v(0) for the flexible missile with rigid-body controller 25
Figure 12. The small motion position phi(0) for the flexible missile with rigid-body controller 26
Figure 13. The large motion position X for the flexible missile with rigid-body controller 27
Figure 14. The large motion position Y for the flexible missile with rigid-body controller 28
Figure 15. The desired and actual trajectory for the flexible missile with rigid-body controller without saturation on the control angle . . 30
Figure 16. The control angle for the flexible missile with rigid-body controller without saturation on the control angle 31
Figure 17. The small motion position v(0) for the flexible missile rigid-body controller without saturation on the control angle 32

Figure 18. The small motion position phi(0) for the flexible missile rigid-body controller without saturation on the control angle 33
Figure 19. The large motion position X for the flexible missile rigid-body controller without saturation on the control angle 34
Figure 20. The large motion position Y for the flexible missile rigid-body controller without saturation on the control angle 35
Figure 21. The desired and actual trajectory for the flexible missile using rigid-body controller with saturation on the control angle 37
Figure 22. The control angle for the flexible missile using rigid-body controller with saturation on the control angle 38
Figure 23. The small motion position v(0) for the flexible missile using rigid-body controller with saturation on the control angle 39
Figure 24. The small motion position phi(0) for the flexible missile using rigid-body controller with saturation on the control angle 40
Figure 25. The large motion position X for the flexible missile using rigid-body controller with saturation on the control angle 41
Figure 26. The large motion position Y for the flexible missile using 42
Figure 27. The desired and actual trajectory for the flexible missile using rigid-body controller with increased speed 43
Figure 28. The control angle for the flexible missile using rigid-body controller with increased speed 44
Figure 29. The small motion position v(0) for the flexible missile using rigid-body controller with increased speed. 45
Figure 30. The small motion position phi(0) for the flexible missile using rigid-body controller with increased speed 46
Figure 31. The large motion position X for the flexible missile using rigid-body controller with increased speed 47
Figure 32. The large motion position Y for the flexible missile using rigid-body controller with increased speed 48
Figure 33. The Bar in Flexure 50
Figure 34. Free Body Diagram Corresponding to a Bar Element 51

ACKNOWLEDGEMENTS

I would like to extend my sincerest thanks and appreciation to my advisor, Professor Liang-Wey Chang for his steadfast support, guidance and always open door policy. I would also like to thank Professor Harold A.Titus and Laboratory Technician Colin Cooper for their technical support.

I would especially like to recognize the accomplishments of my wife, Ulfet. Her steadfast devotion, encouragement, support and confidence, permitted my absence during hours of derivations and computer work while she carried the burden of household and family work.

I. INTRODUCTION

A. BACKGROUND

With the introduction of long slender missiles such as the Vanguard, the Redstone, and various ballistic missiles, the problem of the structural flexibility became severe [Ref. 1]. Due to the limited thrust available from rocket engines, these missiles had to be as light as possible.This meant a sacrifice in structural rigidity.

Missile flexure causes additional aerodynamic loads which in turn cause additional flexure. Coupling occurs between the elastic modes and the control system as the control system gyros sense the flexure motion and the rigid body motion. It has become necessary to actively control the flexible structure and thereby reduce the structural loads and improve the vehicle response such as position, velocity and acceleration. Reduced structural loads will also offer potential for reduced bending stress and fatigue problems.

The objective of this thesis is to develop a dynamic model for a flexible missile and study the dynamic behavior of the flexible missile. Simulation is a valuable tool in the design of new missile systems and in the modification or evaluation of existing systems. A missile simulation allows the engineer to evaluate his design without the expense of actually building and flying the actual missile. System dynamics can be investigated through simulation with a substantial savings in time and expense [Ref. 2].

B. LITERATURE REVIEW

Flexible missile modeling centers on the relationship between the large, rigid-body motion and the small motions due to structural flexibility.

Jenkins [Ref. 2] expresses some techniques used in deriving the equations of motion of a rigid missile for a six degree-of-freedom (6-DOF) simulation. The rigid missiles are characterized by their larger size and low ratio of payload to total weight. Rigid missile dynamic equations were developed using the Newton-Euler Method. The moments and forces along with the mass and moments of inertia are assumed to be known in the body coordinate frame. The transformation between global and local coordinate frame is achieved with a non-homogenous coordinate transformation matrix [Ref. 3: pp. 342]. The resulting rigid-body equations of motion produced the understanding for the derivation of the dynamic equations of the flexible missile.

The Equivalent Rigid Link System [Ref. 4] describes the large-motion kinematics of the system by the ERLS motion and the small motion relative to the ERLS. The application of the finite element techniques and Lagrangian dynamics produces two sets of coupled, nonlinear, ordinary differential equations of motion, of which one set is for the large motions and the other set for the small motions. The small motion is described by the superposition of vibration modes. The modes of the vibration of the flexible bar was derived with the simple-beam theory [Ref. 5: pp.221]. In simple beam theory, it is assumed that the rotation of the element is insignificant compared to the vertical translation and the shear deformation is small relative to the bending deformation. This assumption is valid if the ratio between the length of the bar and its height is relatively large (more than 10). The set of large motion equations are non-linear in both the large and small motion variables while the set of small motion equations are linear in the small motion variable and non-linear in the large motion variables. A solution technique called the Sequential Integration Method [Ref. 6] was developed which allows efficient simulations of systems with inertia coupled motions having non-linear slow motion (large motion) with linear fast motion (small motion). The ERLS model presents a complete, efficient dynamic model able to describe large motion, small motion and their coupling.

An ERLS model of a flexible spacecraft boom was developed and a computer simulation was performed [Ref. 7]. The equations of motion were solved using the Sequential Integration Method. A spatial finite discretization of the boom structure and the application of an assumed polynomial modal response were utilized in the approximate solution to the equations of motion. This work was the basis work for the modeling of the flexible missiles.

Ganon [Ref. 8] performed an experimental validation of the ERLS dynamic model in a vertical plane motion. The small motion was modeled using a shape matrix derived from superposition of natural modes. The agreement between the simulation results and the experimental results justify the application of the ERLS using a natural-mode shape matrix to model the dynamic response of flexible missile.

C. OUTLINE

In this study, the ERLS dynamic model is used to derive the system equations of motion for a flexible missile in 2-D motion. Dynamic response is predicted by solving the equations of motion using the Sequential Integration Method. The application of an assumed natural mode shape and the spatial finite element discretization of the missile provide an approximate solution. Computer simulation for the flexible missile is performed using MATLAB on the MACINTOSH computer. A rigid body controller is included in the simulation to control the pitch angle of the flexible missile.

The ERLS dynamic model of the flexible missile is presented in Chapter Two. A rigid body controller for the flexible missile is described in Chapter Three. The computer simulation methodology and simulation results are presented in Chapter Four. The conclusions and recommendations for future work are presented in Chapter Five.

II. MODEL FORMULATION FOR A FLEXIBLE MISSILE

A. KINEMATICS

In our model, 2-D inertial reference frame, i.e., X and Y, is used for the global motion as shown in Fig.1. The body-fixed coordinate frame, i.e., x and y, is selected to describe the missile local motion.

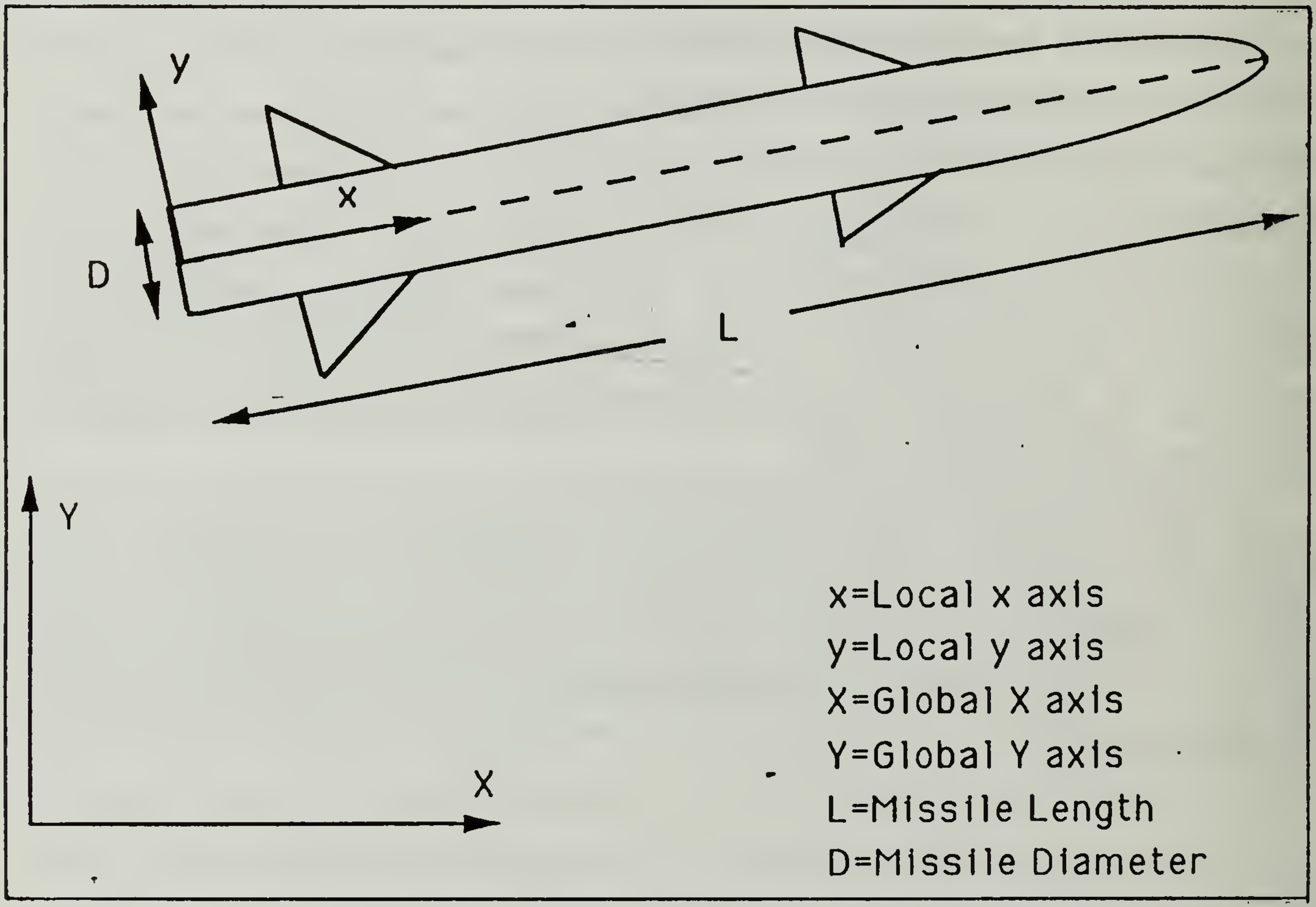

Figure 1. **The general configuration of the missile**

The following assumptions were made for the flexible missile:

(1) Material density is constant throughout the body and the steel was chosen for modeling.

(2) A uniform circular cross section is assumed.

The geometric configuration parameters and material properties of the flexible missile are listed as

Diameter = 0.12 meter

Length = 4.0 meter

Material Density = 7861.05 kg/m^3

Young's Modulus = 2.0 x 10^{11} pascal

The concept of the ERLS is applied to model the kinematics of flexible missile. The main idea of the Equivalent Rigid Link System (Fig.2) is to separate the kinematics of the flexible body into large and small motions.

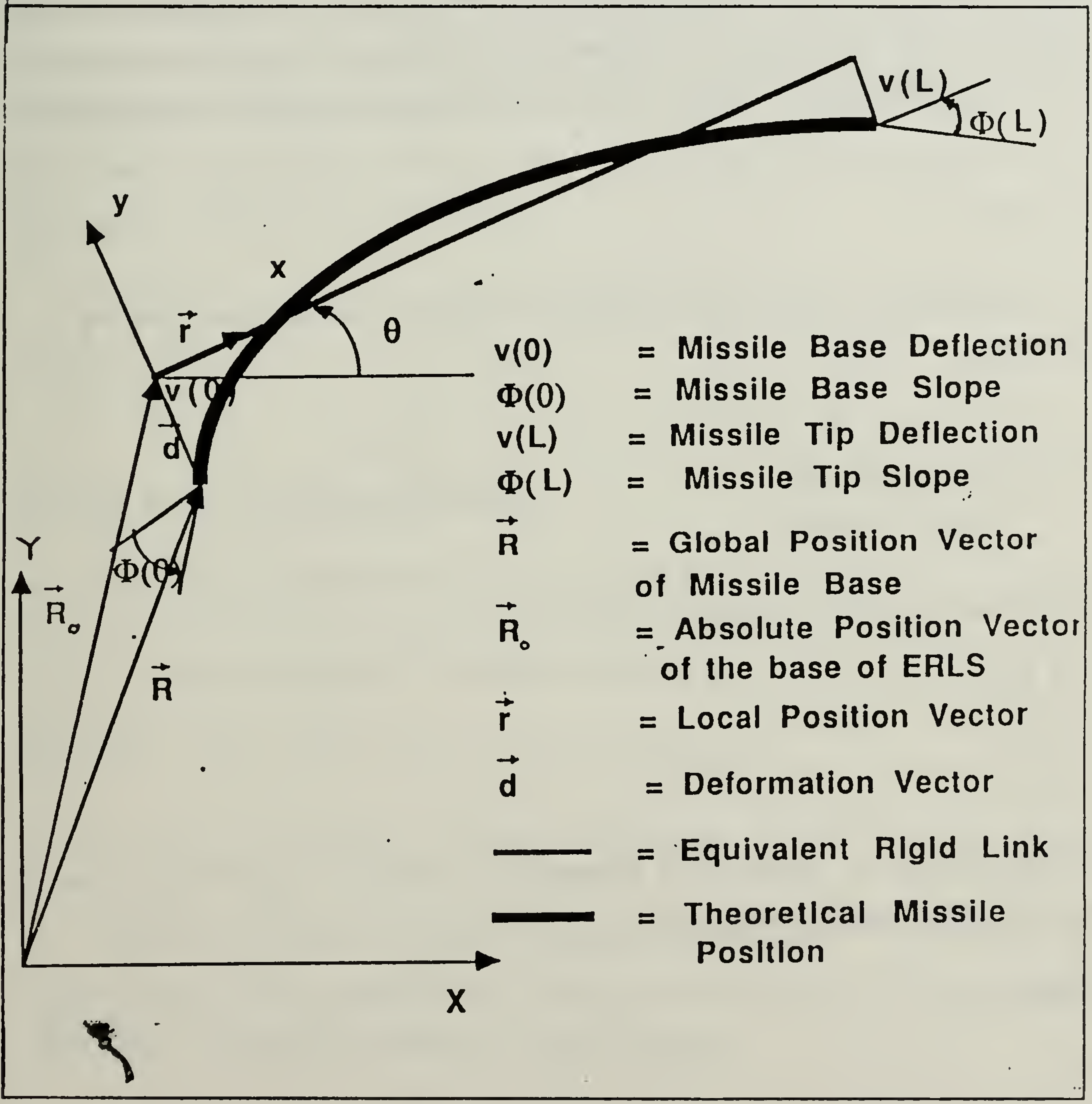
v(L)
Φ(L)
y
x
$\vec{r}$
θ
v(0)
$\vec{d}$
Y
$\vec{R}_o$
Φ(0)
$\vec{R}$
X
v(0) = Missile Base Deflection
Φ(0) = Missile Base Slope
v(L) = Missile Tip Deflection
Φ(L) = Missile Tip Slope
$\vec{R}$ = Global Position Vector of Missile Base
$\vec{R}_o$ = Absolute Position Vector of the base of ERLS
$\vec{r}$ = Local Position Vector
$\vec{d}$ = Deformation Vector
= Equivalent Rigid Link
= Theoretical Missile Position

Figure 2. Equivalent Rigid Link System (ERLS)

The large motion of the missile will be represented by missile base position in X direction X_0, missile base position in Y direction Y_0, and missile pitch angle θ. The small motions resulting from flexible motion are measured relative to the local coordinate frame x, y. v(0) and Φ(0) are the nodal displacement and slope of the missile base respectively. The absolute (global) position of a point on the flexible missile is obtained from coordinate transformations. The global position ($\vec{R}$) of any point position can be defined using ERLS in terms of a local position vector ($\vec{r}$), a deformation vector ($\vec{d}$), and a coordinate transformation matrix (W), i.e.,

$$\vec{R} = W(\vec{r} + \vec{d}) \qquad (2-1)$$

The transformation matrix (W) between the large motion and small motion coordinate is

$$W = \begin{bmatrix} 1 & 0 & 0 \\ X_0 & \cos(\theta) & \sin(\theta) \\ Y_0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \qquad (2-2)$$

and the local rigid body position vector is

$$\vec{r} = \begin{bmatrix} 1 \\ x \\ 0 \end{bmatrix} \qquad (2-3)$$

The deformation vector ($\vec{d}$) is expressed in terms of a nodal displacement vector $\vec{U}$ and a shape function ϕ as

$$\vec{d} = \phi\vec{U} \qquad (2-4)$$

where

$$\vec{U} = [v(0) \quad \Phi(0)]^T \qquad (2-5)$$

The shape function ϕ was derived utilizing a natural-mode superposition and a finite element concept. The Finite Element Method (FEM) was utilized to discretize the flexible body displacements and assigning the nodes. Displacements are for each point along the missile, a function of location and time, and it is necessary to discretize the deformations to obtain an ordinary differential equation. The natural mode shape function of a beam is used to represent the flexural motion of the flexible missile. Only the first two mode shapes are used. The flexible missile is modeled as a continuous Euler-Bernoulli free-free beam, neglecting shear deformation and rotary inertia effects. The nodal points are the base points of the flexible missile. Appendix A shows the mode shape function derivation. In this case, ϕ is found as,

$$\phi = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ a & b \end{bmatrix} \qquad (2-6)$$

where a and b are defined as following,

$$a = F_1(C_1(\cos\beta_1 x + \cosh\beta_1 x) + (\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_3(C_2(\cos\beta_2 x + \cosh\beta_2 x) + (\sin\beta_2 x + \sinh\beta_2 x)) \qquad (2-7)$$

$$b = F_2(C_1(\cos\beta_1 x + \cosh\beta_1 x) + (\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_4(C_2(\cos\beta_2 x + \cosh\beta_2 x) + (\sin\beta_2 x + \sinh\beta_2 x)) \qquad (2-8)$$

Lagrangian dynamics is used to acquire the equations of motion and the kinetic energy of the system will be needed in the development of the Lagrangian expression. The absolute velocity is listed as follows,

$$\dot{\vec{R}} = \dot{W}(\vec{r} + \vec{d}) + W\dot{\vec{d}} \qquad (2-9)$$

B. KINETICS

Lagrangian Dynamics is a systematic way to derive equations of motion for complex systems like flexible missiles. Lagrange's equation is written as,

$$\frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{q}_i}\right) - \frac{\partial KE}{\partial q_i} + \frac{\partial PE}{\partial q_i} = Q_i \qquad (i = 1,2,\ldots,n) \qquad (2-10)$$

where

K.E. = Kinetic Energy

P.E. = Potential Energy

q_i = Generalized coordinates

Q_i = Generalized forces

n = Number of degrees of freedom

The generalized coordinates are chosen to be,

$$\vec{q} = [X_0 \;\; Y_0 \;\; \theta \;\; v(0) \;\; \Phi(0)]^T \qquad (2-11)$$

The kinetic energy of the system is defined as follows,

$$K.E. = \frac{1}{2}\int \dot{\vec{R}}^T \dot{\vec{R}}\, dm \qquad (2-12)$$

By substituting Eq.(2-1) into Eq. (2-12), the kinetic energy is written as

$$KE = \frac{1}{2}\int (\vec{r}^T \dot{W}^T \dot{W} \vec{r} + \vec{2r}^T \dot{W}^T \dot{W} \phi \vec{U} + \vec{2r}^T \dot{W}^T W \phi \dot{\vec{U}}$$

$$+ \vec{U}^T \phi^T \dot{W}^T \dot{W} \phi \vec{U} + 2U^T \phi^T \dot{W}^T W \phi \dot{\vec{U}} + \dot{\vec{U}}^T \phi^T W^T W \phi \vec{U}^T) dm \qquad (2-13)$$

The potential energy of the system includes the strain energy of the flexible missile and the gravitational potential energy.

$$P.E. = \frac{1}{2}\int \vec{U}^T \Gamma^T C \Gamma \vec{U} dx - \int \vec{R}^T \vec{g}\, dm \qquad (2-14)$$

where

Γ = Spatial derivative of the shape function

C = Rigidity Matrix

$\vec{g}$ = Gravitational acceleration vector

Appendix B includes the development of the equations of motions using Lagrange equations.

Generalized forces will be found by virtual work principle. It is assumed that the only force is the thrust force which is applied to the base of the flexible missile as shown in Fig.3. Other forces like aerodynamic and damping forces are neglected.

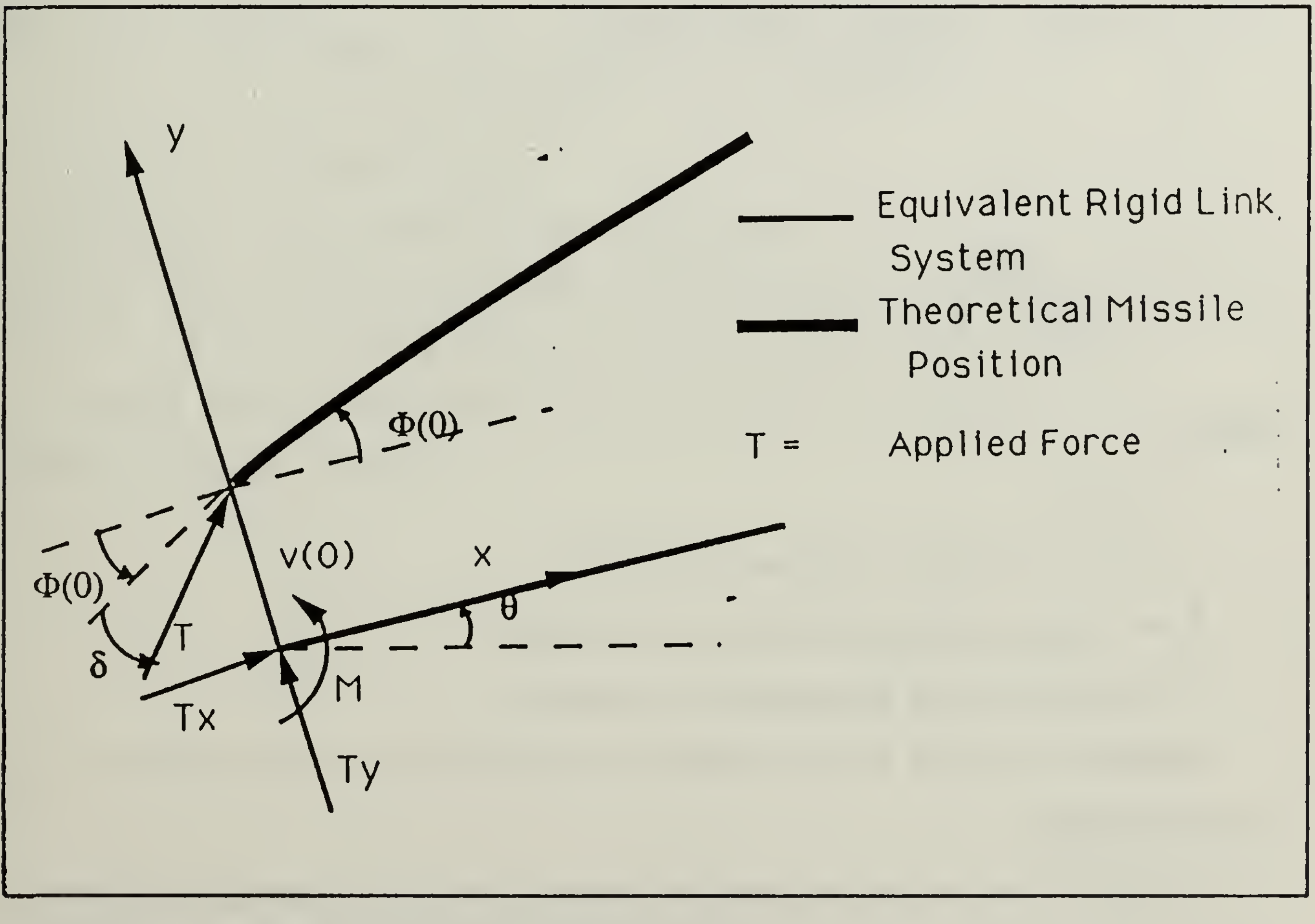

Figure 3. Applied Forces on Flexible Missile

The thrust force is composed of two components i.e.,

$$T_x = T\cos(\delta + \Phi(0)) \qquad (2-15)$$

$$T_y = T\sin(\delta + \Phi(0)) \qquad (2-16)$$

and a moment applied on the missile is,

$$M_\theta = -T\cos(\delta + \Phi(0))v(0) \qquad (2-17)$$

By applying virtual work principle, the generalized forces of flexible missile is found below.

$$\vec{F}_\xi = \begin{bmatrix} T\cos(\theta) - T\delta\sin(\theta) - T\Phi(0)\sin(\theta) \\ T\sin(\theta) + T\delta\cos(\theta) + T\Phi(0)\cos(\theta) \\ -Tv(0) \end{bmatrix} \qquad (2-18)$$

$$\vec{F}_u = \begin{bmatrix} T\delta + T\Phi(0) \\ -Tv(0) \end{bmatrix} \qquad (2-19)$$

where

T = Force

δ = Control Angle being assumed small

$\vec{F}_\xi$ = Large motion generalized force vector

$\vec{F}_u$ = Small motion generalized force vector

Appendix C shows the derivation of the generalized forces using the Virtual Work Principle.

The derivation of the equations of motion from the Lagrangian formulation yields two sets of coupled equations. One set describes the large motions and the another set describes the small motions. These two sets of equations are non-linear, coupled, second-order, ordinary differential equations of the form,

$$M_{qq}\,\ddot{\vec{\xi}} + M_{qn}\,\ddot{\vec{U}} = \vec{F}_q \qquad (2-20)$$

$$M_{nq}\,\ddot{\vec{\xi}} + M_{nn}\,\ddot{\vec{U}} + G_n\,\dot{\vec{U}} + K_n\,\vec{U} = \vec{F}_n \qquad (2-21)$$

where:

$M_{qq} \equiv$ 3 x 3 effective inertia matrix for large motions

$M_{qn} \equiv$ 3 x 2 coupled inertia matrix of the small motion effect on large motions

$\vec{F}_q \equiv$ 3 x 1 load vector for the large motions

$M_{nq} \equiv$ 2 x 3 coupled inertia matrix of the large motion effect on small motions

$M_{nn} \equiv$ 2 x 2 effective inertia matrix for small motions

$G_n \equiv$ 2 x 2 gyroscopic matrix

$K_n \equiv$ 2 x 2 stiffness matrix

$\vec{F}_n \equiv$ 2 x 1 load vector for the small motions

$\vec{\xi} \equiv$ 3 x 1 vector, generalized coordinates of the large motions

$\vec{U} \equiv$ 2 x 1 vector, generalized coordinates of the small motions

The detailed development of the equations of motion and definitions of the terms are described in Appendix B.

III. THE DEVELOPMENT OF A RIGID-BODY CONTROLLER

In our research, the controller is developed based on the rigid-body assumption. The purpose of including the rigid-body controller is to perform computer simulation and study the dynamic behavior of the flexible missile. The pitch angle of the missile is controlled by the rigid-body controller which is a single input single output (SISO) controller. A general configuration of the flexible missile and rigid-body controller are shown in Fig.4.

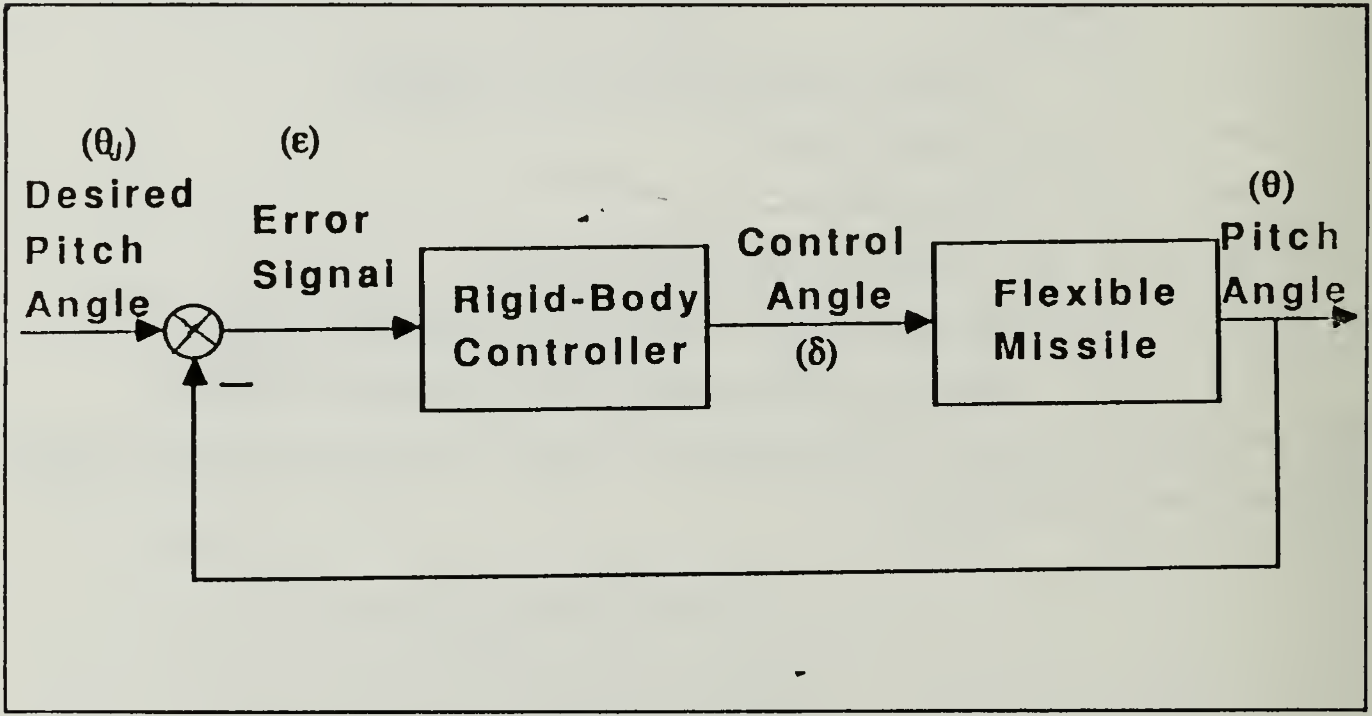

Figure 4. The flexible missile with rigid-body controller

The pitch angle is the control variable and the desired states are desired pitch angle θ_d, desired angular velocity $\dot{\theta}_d$, and desired angular acceleration $\ddot{\theta}_d$. The control angle (δ) is assumed small.

The desired error function is defined as

$$\ddot{\varepsilon} + K_v\dot{\varepsilon} + K_p\varepsilon = 0 \qquad (3-1)$$

where,

K_p = The position feedback gain.

K_v = The velocity feedback gain.

$\varepsilon = \theta - \theta_d$ = The error in the position.

$\dot{\varepsilon} = \dot{\theta} - \dot{\theta}_d$ = The error in velocity.

$\ddot{\varepsilon} = \ddot{\theta} - \ddot{\theta}_d$ = The error in acceleration.

The control design begins with the equation of rigid-body (large motion) that is obtained by modifying and expanding Eq.(2-20) as

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\eta}_1 \\ \ddot{\eta}_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \delta \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad (3-2)$$

Where,

$$M_{11} = \begin{bmatrix} M_{qq}(1,1) & M_{qq}(1,2) \\ M_{qq}(2,1) & M_{qq}(2,2) \end{bmatrix} \qquad (3-3)$$

$$M_{12} = \begin{bmatrix} M_{qq}(1,3) \\ M_{qq}(2,3) \end{bmatrix} \qquad (3-4)$$

$$M_{21} = [M_{qq}(3,1) \quad M_{qq}(3,2)] \qquad (3-5)$$

$$M_{22} = [M_{qq}(3,3)] \qquad (3-6)$$

$$\ddot{\vec{\eta}}_1 = \begin{bmatrix} \ddot{X}_0 \\ \ddot{Y}_0 \end{bmatrix} \qquad (3-7)$$

$$\ddot{\vec{\eta}}_2 = [\ddot{\theta}] \qquad (3-8)$$

$\vec{f}_1$ = Gravity, centrifugal and coriolis forces.

$\vec{f}_2$ = Gravity, centrifugal and coriolis forces.

$$\vec{h}_1 = \begin{bmatrix} T\cos(\theta) \\ T\sin(\theta) \end{bmatrix} \tag{3-9}$$

$$\vec{h}_2 = [0]$$

$$\vec{b}_1 = \begin{bmatrix} -T\sin(\theta) \\ T\cos(\theta) \end{bmatrix} \tag{3-10}$$

$$\vec{b}_2 = [0]$$

Eq.(3-2) can be rewritten in a tensor form as

$$M_{11}\ddot{\vec{\eta}}_1 + M_{12}\ddot{\vec{\eta}}_2 = \vec{f}_1 + \vec{h}_1 + \vec{b}_1\delta \tag{3-11}$$

$$M_{21}\ddot{\vec{\eta}}_1 + M_{22}\ddot{\vec{\eta}}_2 = \vec{f}_2 \tag{3-12}$$

From Eq.(3-11),

$$\ddot{\vec{\eta}}_1 = M_{11}^{-1}\left[\vec{f}_1 + \vec{h}_1 + \vec{b}_1\delta - M_{12}\ddot{\vec{\eta}}_2\right] \tag{3-13}$$

Substituting Eq.(3-13) into Eq.(3-12), we find,

$$A\ddot{\vec{\eta}}_2 = \vec{F} - \vec{B}\delta \tag{3-14}$$

Where,

$$A = M_{22} - M_{21}M_{11}^{-1}M_{12} \tag{3-15}$$

$$\vec{F} = \vec{f}_2 - M_{21}M_{11}^{-1}\left[\vec{f}_1 + \vec{h}_1\right] \tag{3-16}$$

$$\vec{B} = M_{21}M_{11}^{-1}\vec{b}_1 \tag{3-17}$$

Eq.(3-1) can be rewritten as

$$(\ddot{\theta} - \ddot{\theta}_d) + K_v(\dot{\theta} - \dot{\theta}_d) + K_p(\theta - \theta_d) = 0 \qquad (3-18)$$

By substituting Eq.(3-14) into Eq.(3-18), we find the control angle (δ)

$$\delta = -B^{-1}\left[A(\ddot{\theta}_d - K_v(\dot{\theta} - \dot{\theta}_d) - K_p(\theta - \theta_d)) - F\right] \qquad (3-19)$$

The control angle (δ) is a function of the pitch angle (θ) , desired pitch angle (θ_d), angular velocity ($\dot{\theta}$), desired angular velocity ($\dot{\theta}_d$), desired angular acceleration ($\ddot{\theta}_d$), the position feedback gain Kp , the velocity feedback gain K_v, and the matrices i.e., A, B, F. Since the thrust force T is included in F and B, the magnitude of the thrust force will thus affect the control angle δ . K_v and K_p are adjustable to obtain the desired response of the pitch angle θ.

IV. COMPUTER SIMULATION

A. SIMULATION OBJECTIVES

In literature, the application of forces has been limited to gravity, aerodynamic forces and thrust [Ref. 2]. No damping has been implemented. The thrust and aerodynamic forces can be found fairly accurately with standard test and design procedures such as static firings to obtain thrust versus time profiles for the engine and wind tunnel measurements to determine the aerodynamic forces. Gravitational forces can be calculated from the knowledge of the missile's position relative to the earth. The mass including fuel can be estimated from knowledge of the missile's weight before and after burnout (from the measurements), and by using a mathematical relationship (often linear) for the decrease in missile mass over the engine burntime.

In this work, the missile's weight is assumed constant and the aerodynamic forces are zero. The deformations resulting from the structural flexibility have been assumed small and small control angle assumptions are used in the rigid-body controller design.

The primary purpose of this study is to complete the simulation of the flexible missile in 2-D motion, where the bending effect is considered to be the only flexibility source and a rigid-body controller as developed in the last section is included.

B. SOLUTION TECHNIQUE

Many numerical integration techniques can be applied in the solution of the equations of the motion. The main consideration in the selection of the integration technique is the size of the time step necessary to integrate the equations of motions with numerical accuracy and stability.

The type of the equations of motion (2-20, 2-21) in this research permits the application of the Sequential Integration Method [Ref. 6]. The linear equations of small motions are integrated implicitly and the large motion solutions are obtained using explicit integration method. The implicit methods are effective for linear systems with high frequencies and the explicit methods are effective for solving nonlinear systems with low frequencies. The implicit method is especially effective for linear systems having a wide range of frequencies of which only the lower frequencies are excited. The size of the time step need only be chosen to make the solution of the excited modes sufficiently accurate. Explicit methods are effective for large scale systems with low frequencies. Because of the low frequencies, the size of the time step that we choose for stability need not be small. In addition, the explicit method does not need iterative procedures, which are time consuming for non-linear systems.

C. THE COMPUTER SIMULATION CODE

A high level computer language, i.e., MATLAB, was chosen to simulate the missile system. The MATLAB was designed for matrix operations. The simulation code was developed with modular MATLAB routines.

The simulation code can be divided into three levels : LEVEL 1 (an overview) separates the code into primary portions of INITIALIZATION, PLANT DESCRIPTION, INTEGRATION, SYSTEM CONTROL and OUTPUT. LEVEL 2 facilitates several subroutines for LEVEL 1. A listing of MATLAB routines required for manipulation in LEVEL 2 is placed in LEVEL 3.

D. SIMULATION RESULTS

The computer simulation was performed with variable parameters which determine missile speed and controller bandwidth. These parameters include force T, K_v and K_p.

The simulation outputs will be presented in large motions, small motions and control angle. The large motions are X_0, Y_0 and θ and the small motions are v(0) and $\Phi(0)$. The initial condition for all runs was the pitch angle of 45°. The simulation work was divided into two areas. First a simulation was performed for the rigid missile using the rigid-body controller. These results were used as a baseline for comparison with the results of the flexible missile using the rigid-body controller system. Second a simulation was performed for the flexible missile and rigid-body controller system. Only the trajectory control will be discussed in the analyze of the dynamics of the flexible missile. The desired trajectory was specified as

$$\theta = \begin{bmatrix} 45^\circ & when\ 0.1 \leq t < 0.1 \\ 45^\circ - 112.5t & when\ 0.1 \leq t < 0.4 \\ 0 & when\ t \geq 0.4 \end{bmatrix}$$

The control angle is assumed to be limited to +/-10 degrees (small angle). This restriction puts a saturation line to the control input.

Fig.5 represents the graphical results of the desired and actual trajectory for the rigid missile and rigid-body controller. The force (T) and controller bandwidth (ω_n) were 30000 N and 3 rad/s respectively. The dark black line and dashed line represent the desired and actual trajectory, respectively. The control angle (δ) is shown in Fig.6 and Figures 7-8 present the base positions ($X_{0,} Y_0$) in large motion.

After presenting results for the rigid missile with rigid-body controller, the dynamic behavior of the flexible missile with rigid-body controller will be studied next. The force (T) and controller bandwidth (ω_n) again were 30000 N and 3 rad/s, respectively. Fig.9 shows the desired and actual trajectory. The control angle is presented in Fig.10. After 0.1 sec, the missile initially needs a control angle which is less than 10 degrees. The effects of small motion can be seen on control angle clearly. The base deflection and slope are shown in Figures 11-12. There is no small motion effects on flexible missile between 0-0.1 sec because of zero control angle and no force components. After 0.1 sec, the small motions are excited and have an amplitude of 10^{-3}. Figures 13-14 present the base positions of the flexible missile in large motion. The large motion behaves like the rigid-body motion, which implies that the coupling effect between the large motion and small motion is small. The dynamic behavior of the flexible missile is dominant by the large motion in this case since the bandwidth of the controller is low and small motion is not significantly excited.

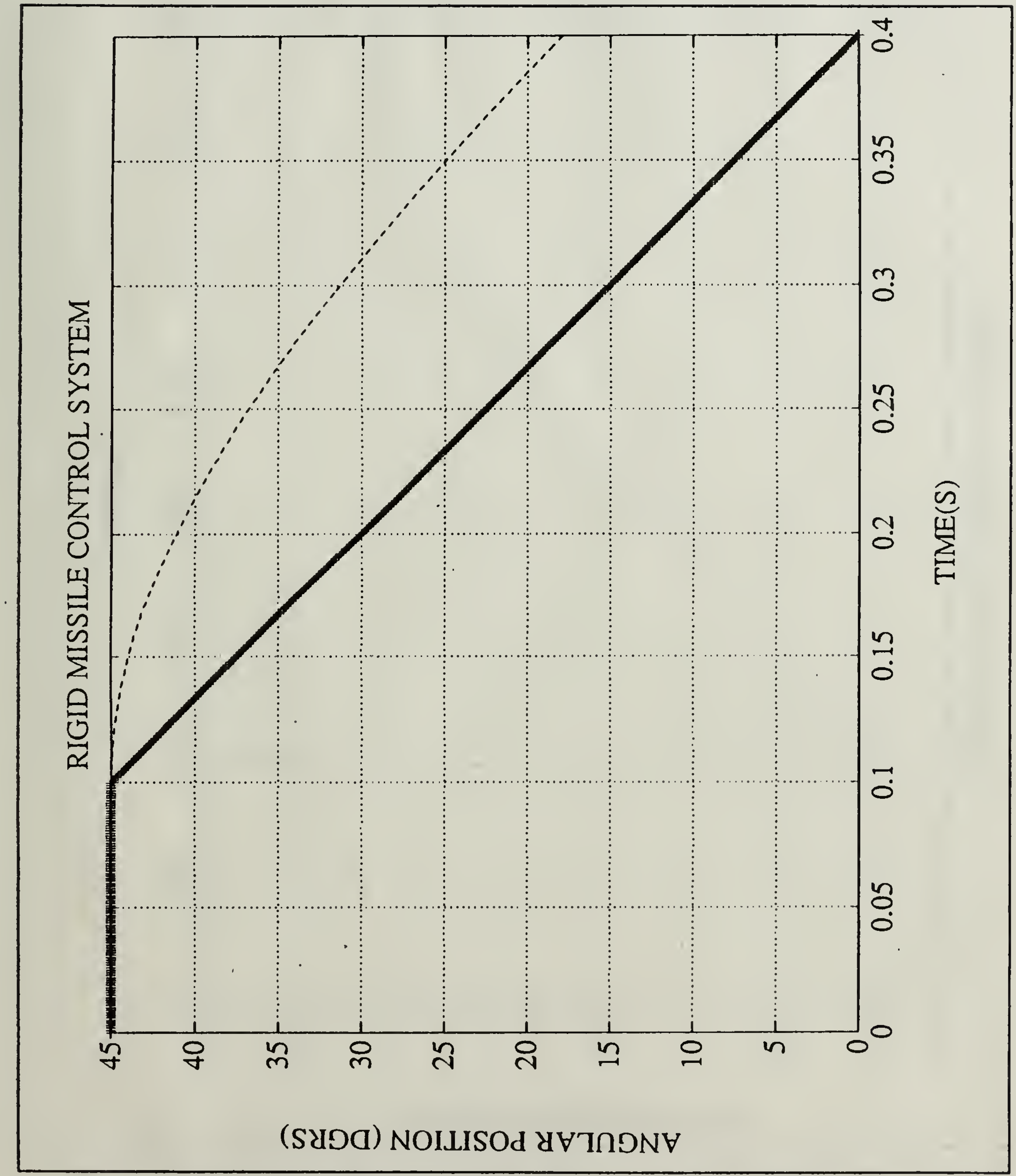

Figure 5. The desired and actual trajectory for the rigid missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

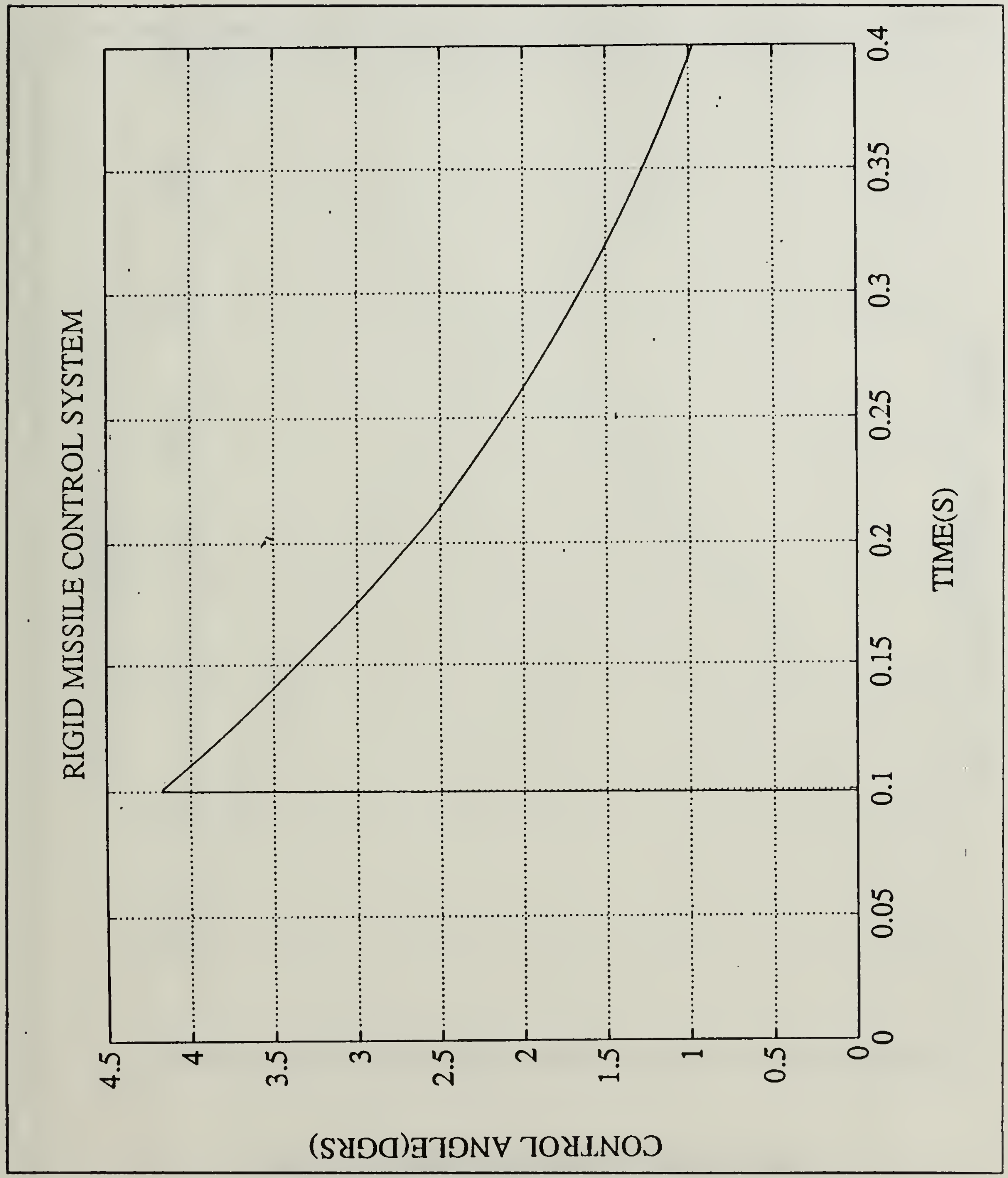

Figure 6. The control angle for the rigid missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

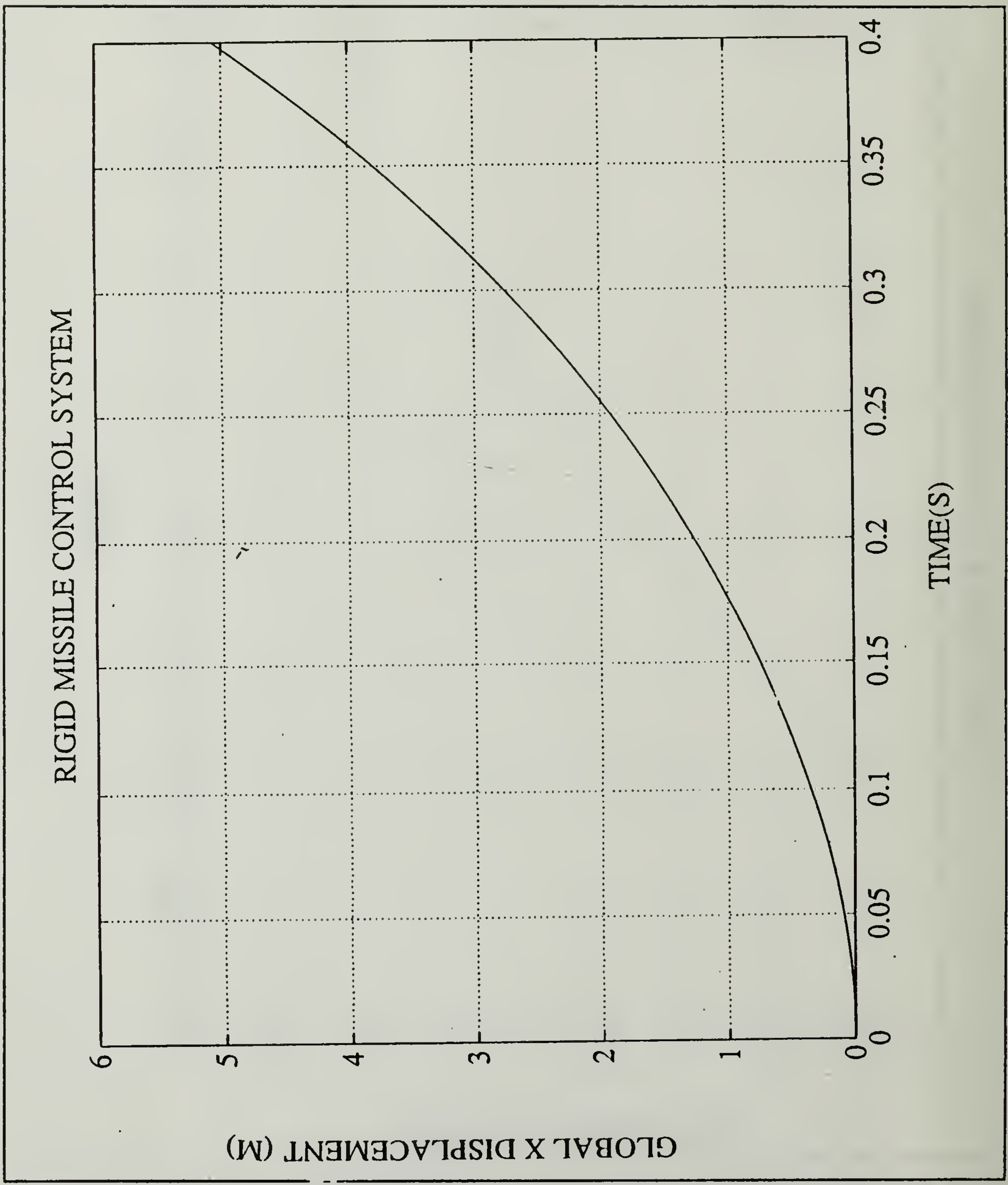

Figure 7. The large motion position X for the rigid missile with rigid-body controller ($T = 30000$ N, $\omega_n = 3$ rad/s)

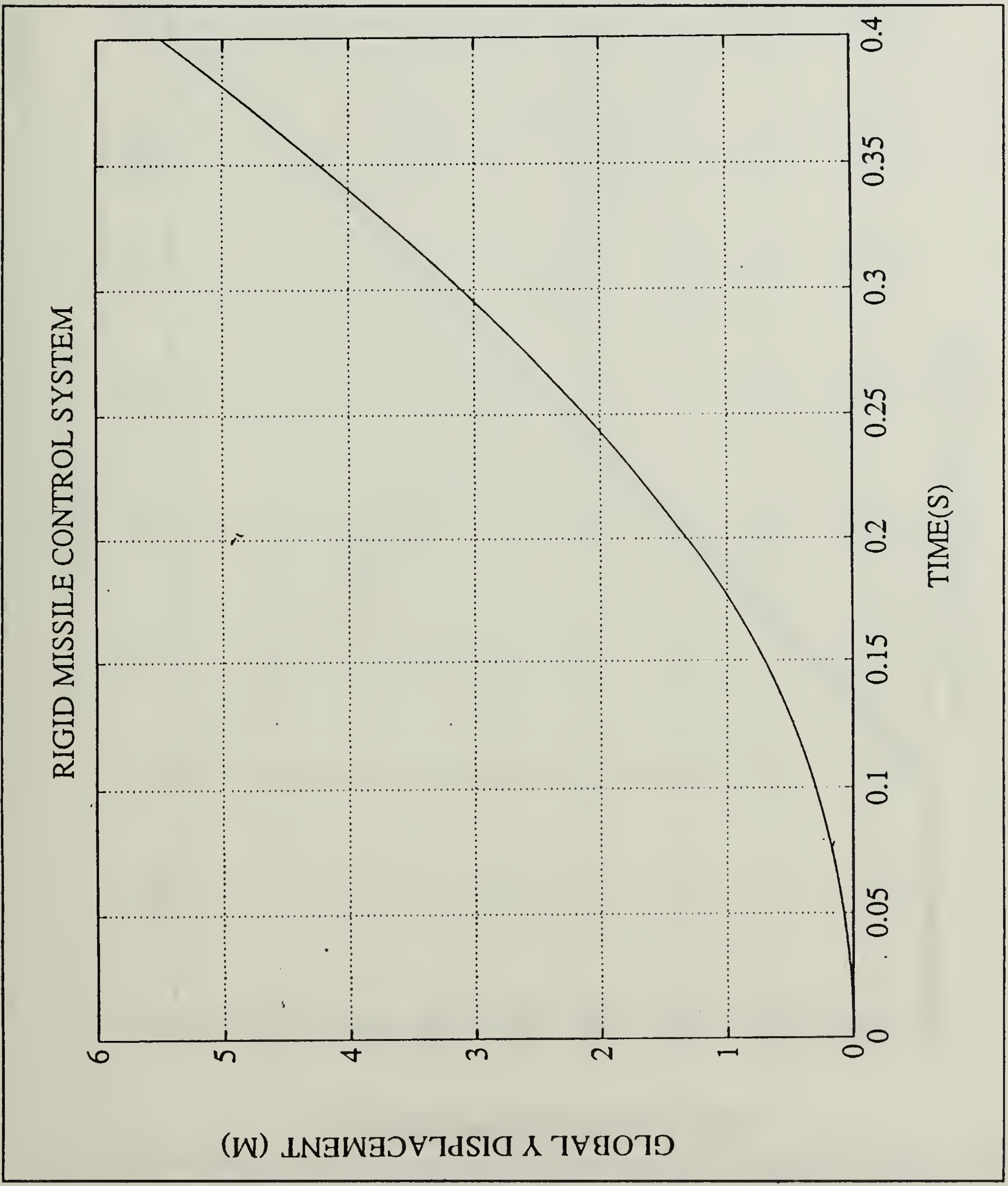

Figure 8. The large motion position Y for the rigid missile with rigid-body controller ($T = 30000$ N, $\omega_n = 3$ rad/s)

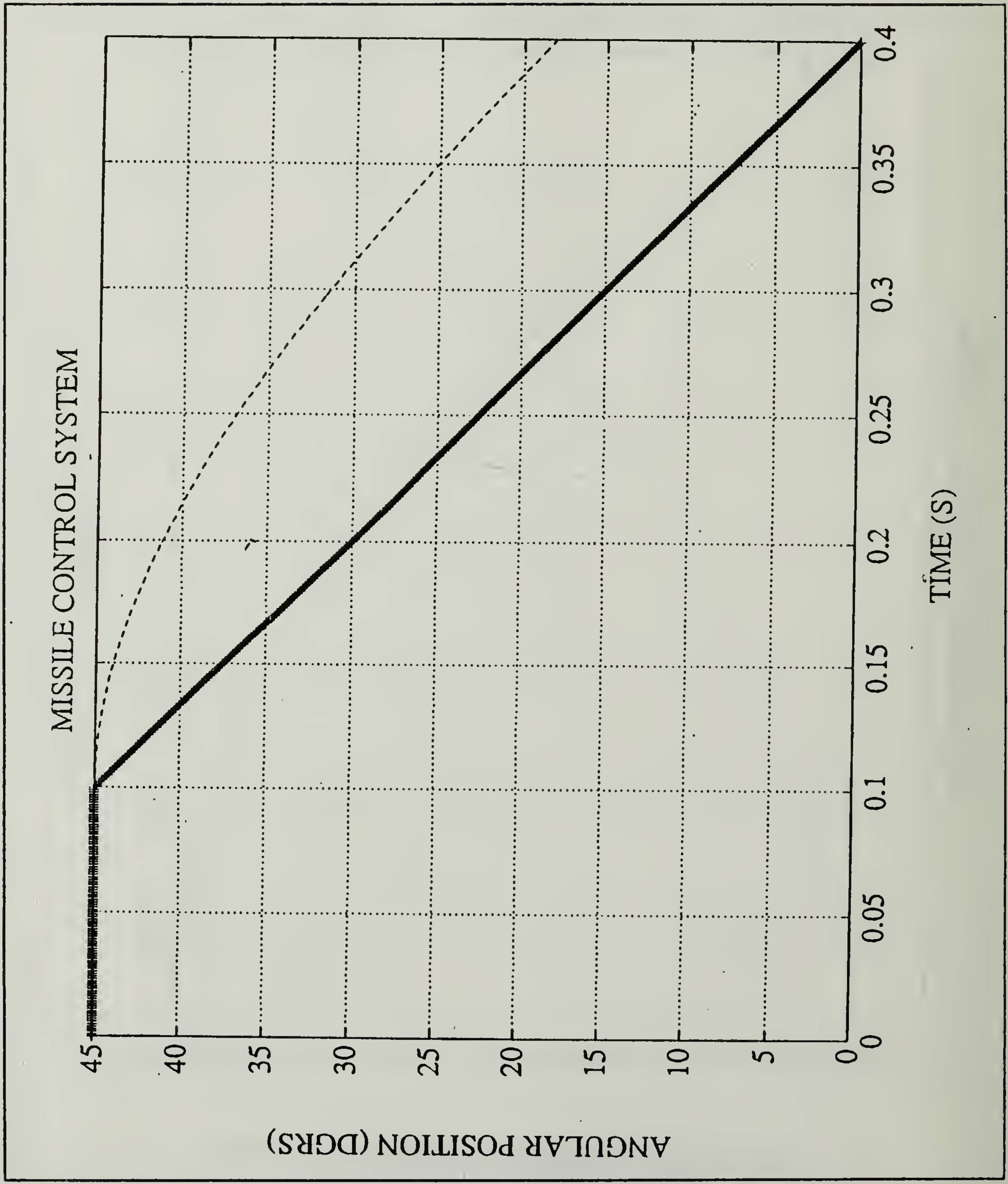

Figure 9. The desired and actual trajectory for the flexible missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

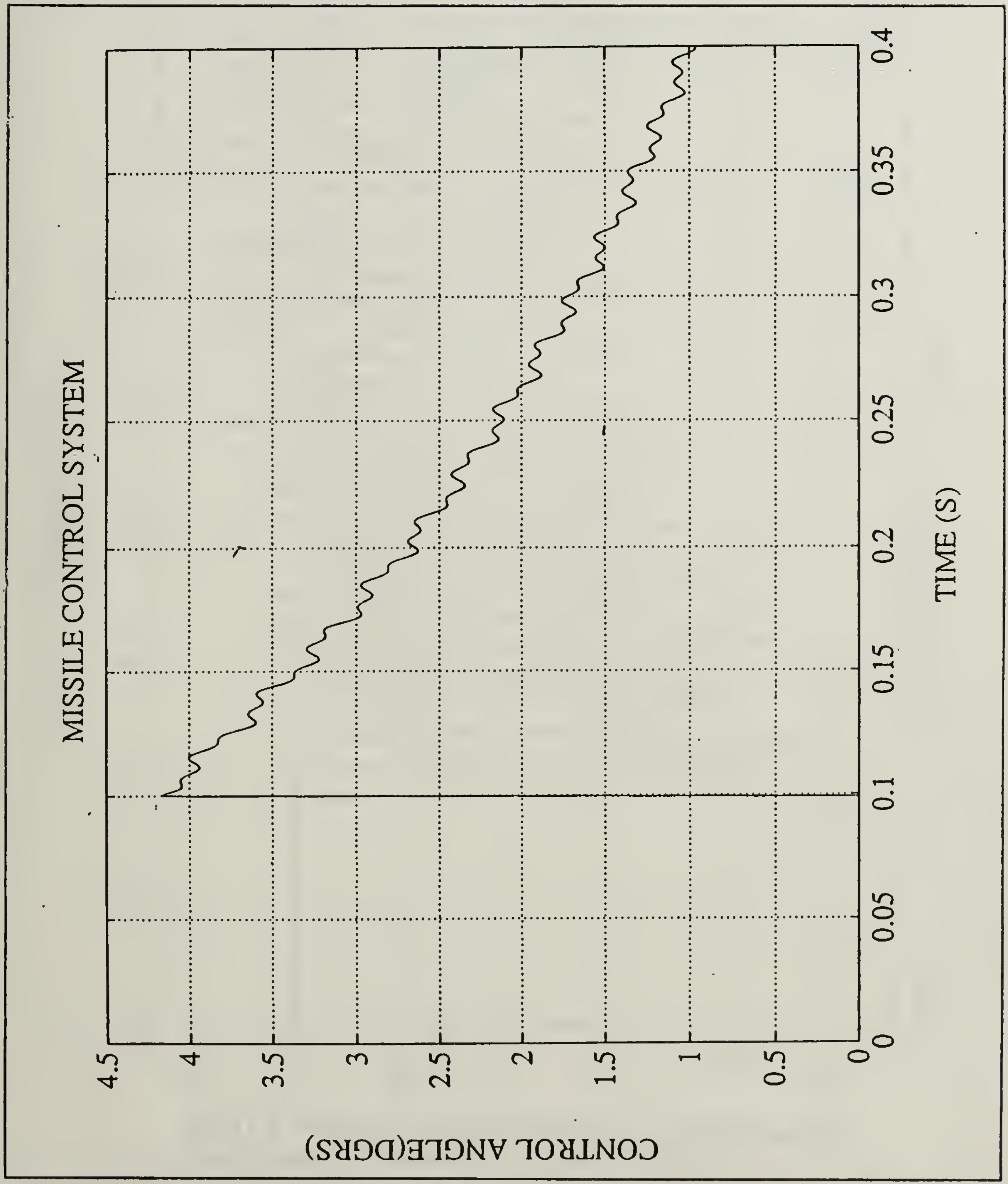

Figure 10. The control angle for the flexible missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

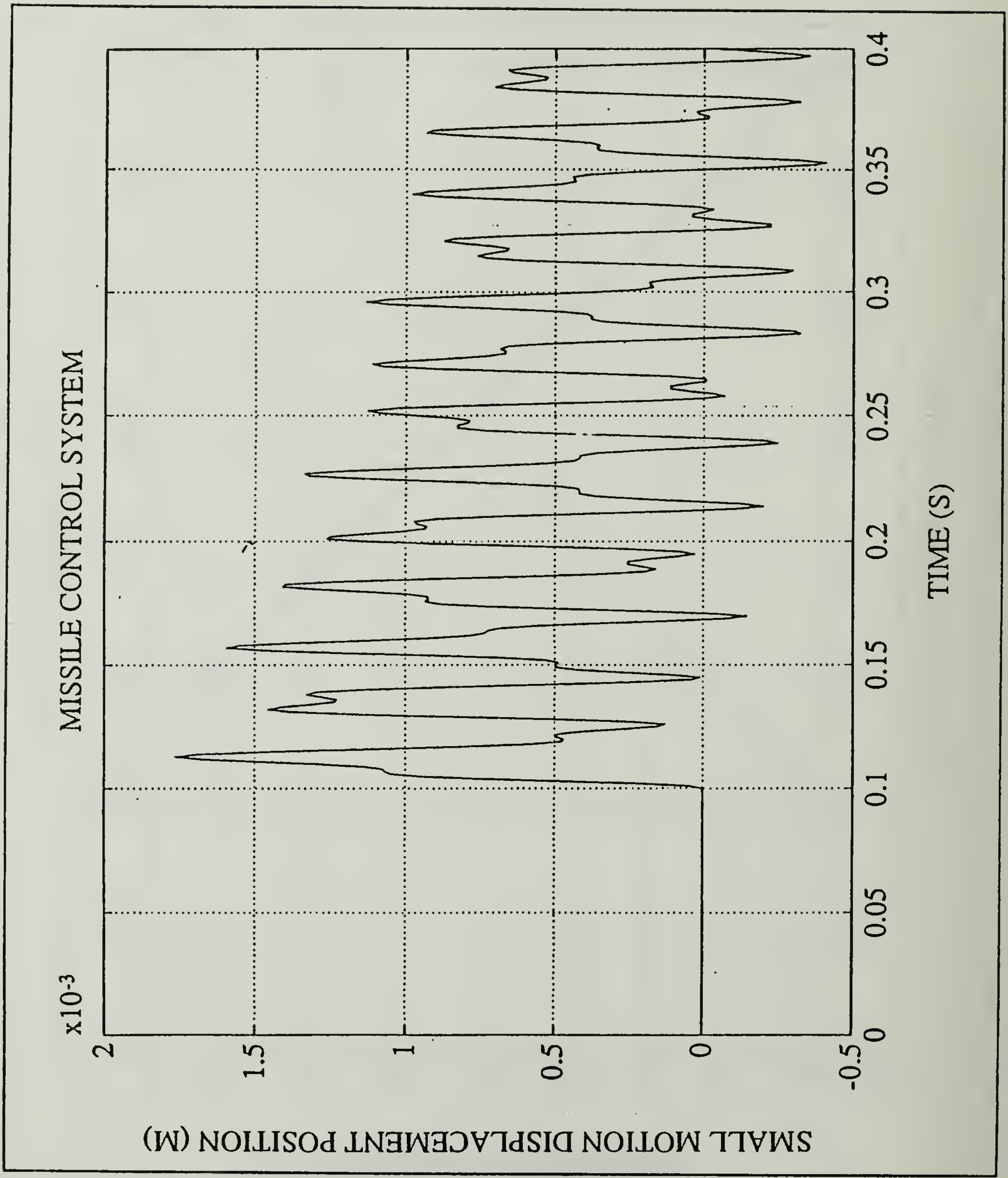

Figure 11. The small motion position v(0) for the flexible missile with rigid-body controller.(T = 30000 N, ω_n = 3 rad/s)

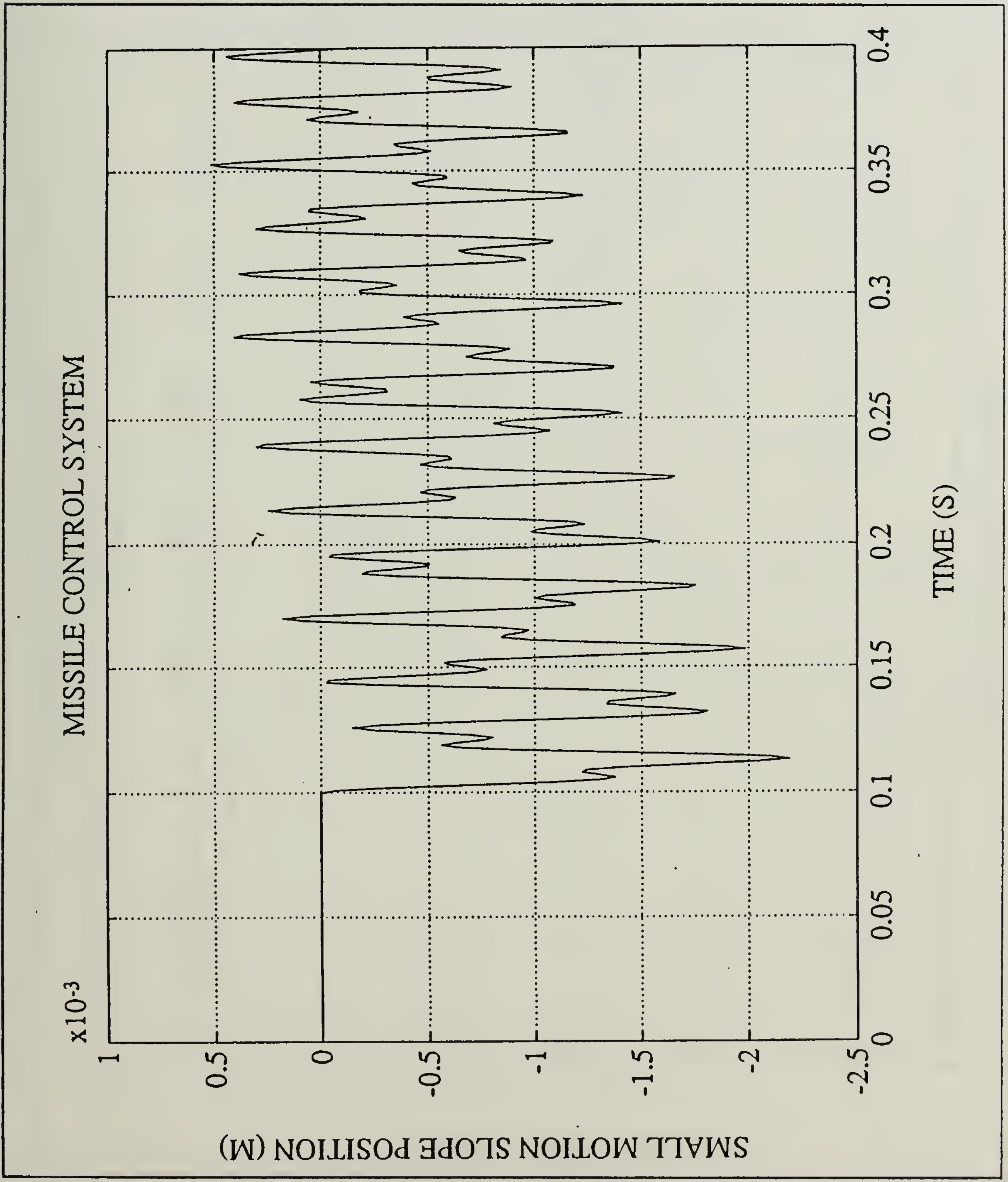

Figure 12. The small motion position phi(0) for the flexible missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

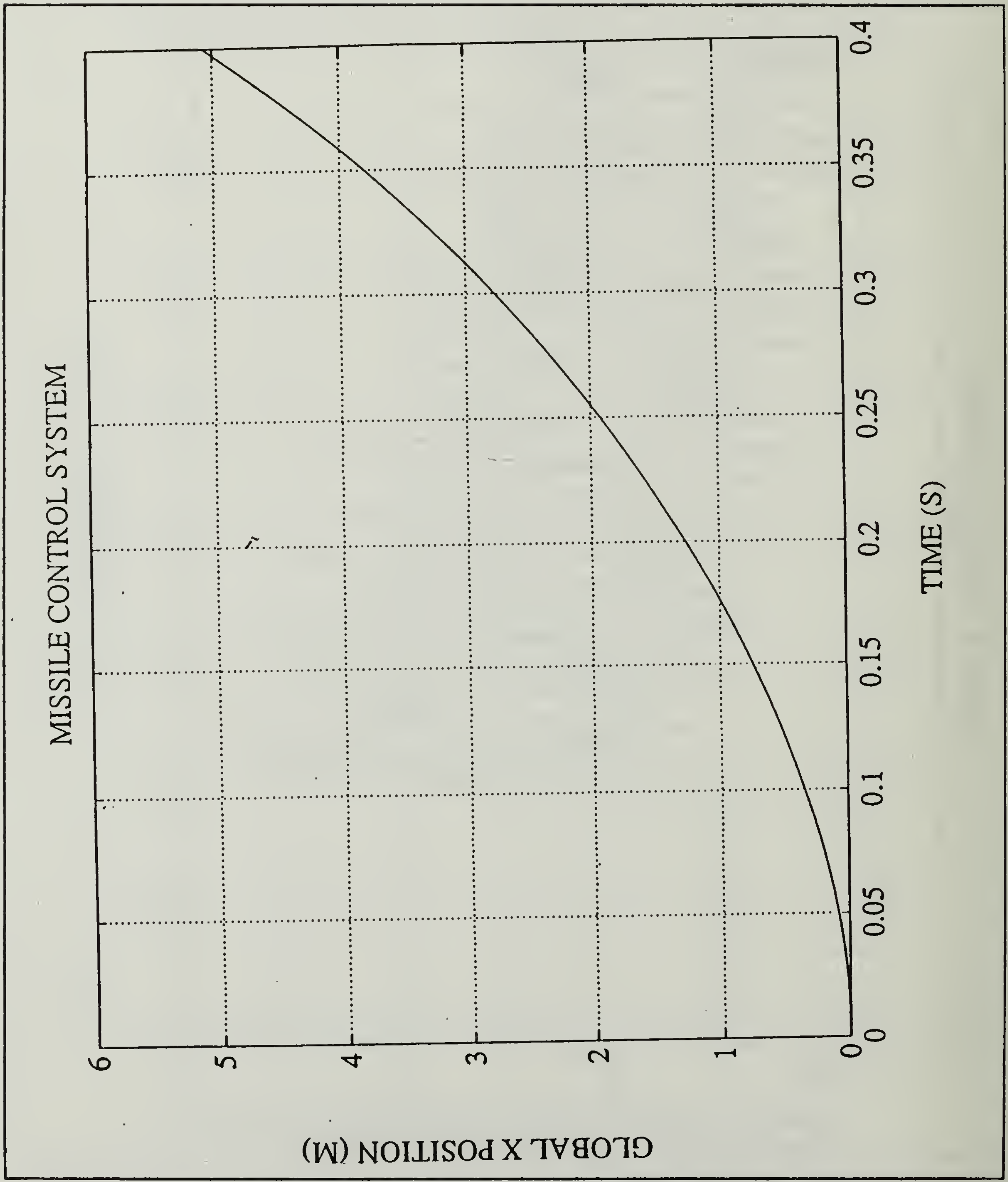

Figure 13. The large motion position X for the flexible missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

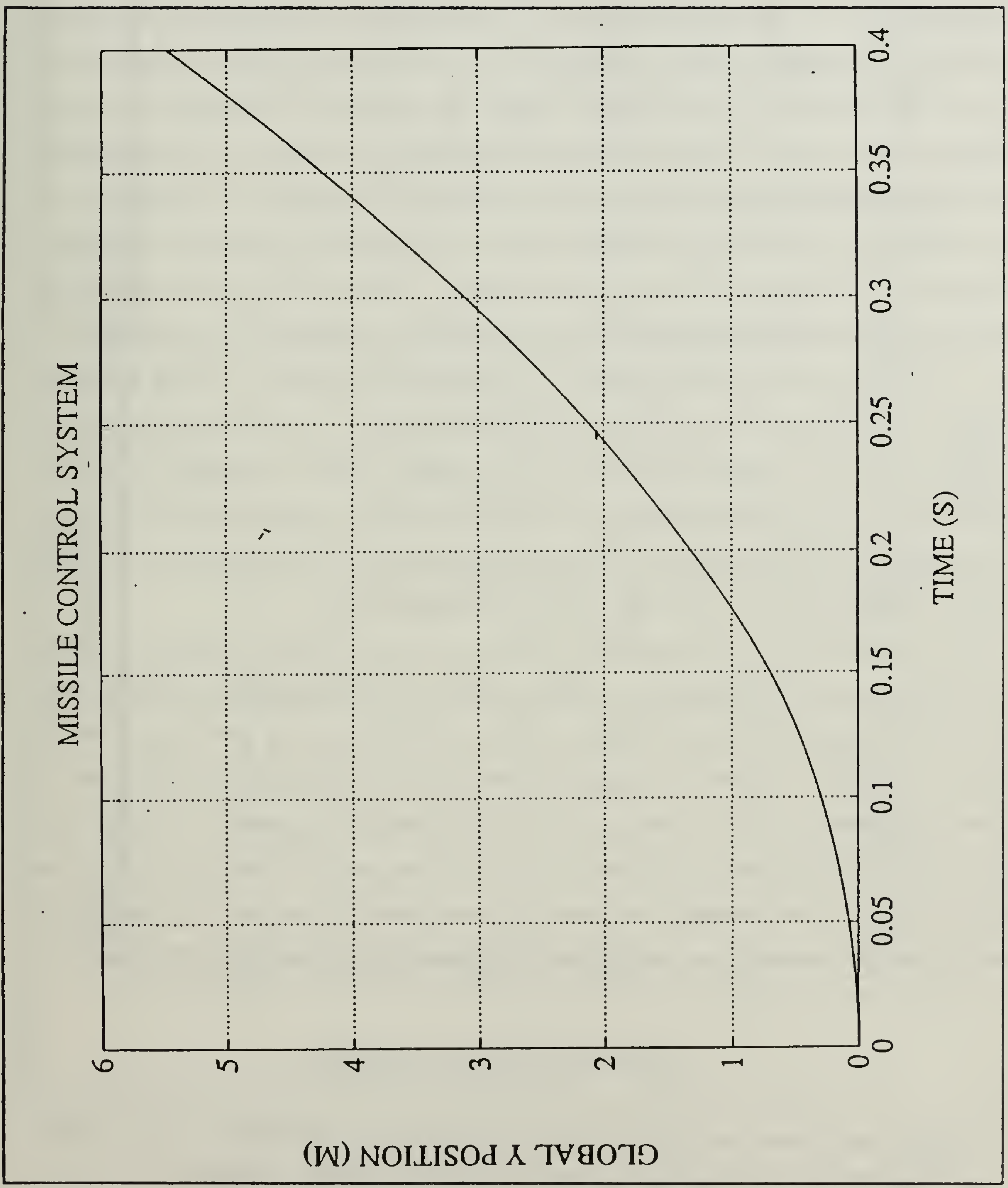

Figure 14. The large motion position Y for the flexible missile with rigid-body controller (T = 30000 N, ω_n = 3 rad/s)

Two kinds of tests were performed to further the study of the dynamic behavior of the flexible missile.

The first test was to study dynamics due to the increase of the control system bandwidth (ω_n). The desired and actual trajectory of the flexible missile with rigid-body controller without saturation on the control angle is presented in Fig.15. The force (T) is still 30000 N while the controller bandwidth (ω_n) is increased to 300 rad/s. As the controller bandwidth is increased, the gap between controller bandwidth and system natural frequency is smaller. The system will be unstable at controller bandwidths close to the system fundamental frequency which can be as high as 270 rad/s in this case. Note that the fundamental frequency of the simple beam with free-free boundary conditions was calculated as $\omega_{n1} = 209.2053$ rad/s, and the missile's fundamental frequency will be increased due to the coupling between large and small motion. The control dynamics interfere with the structural dynamics. The higher control frequency with the bandwidth of 300 rad/s significantly excites the small motion of the structure. The graphs of control angle and the small motions and large motions of the missile base (Figures 16-17-18-19-20) show the unstable state.

The bandwidth of the controller must be set much lower than the fundamental frequency of the missile in order to use the rigid-body control. This implies that the pitch-angle response speed, which is determined by the control bandwidth is limited. To keep the response speed, the missile structure must be designed sufficiently rigid to possess a high fundamental frequency. On the other hand, the missile payload will affect the natural frequency of the missile structure with the heavier the payload, the lower the fundamental frequency of the missile. Therefore, the payload must be limited to achieve high-speed control response.

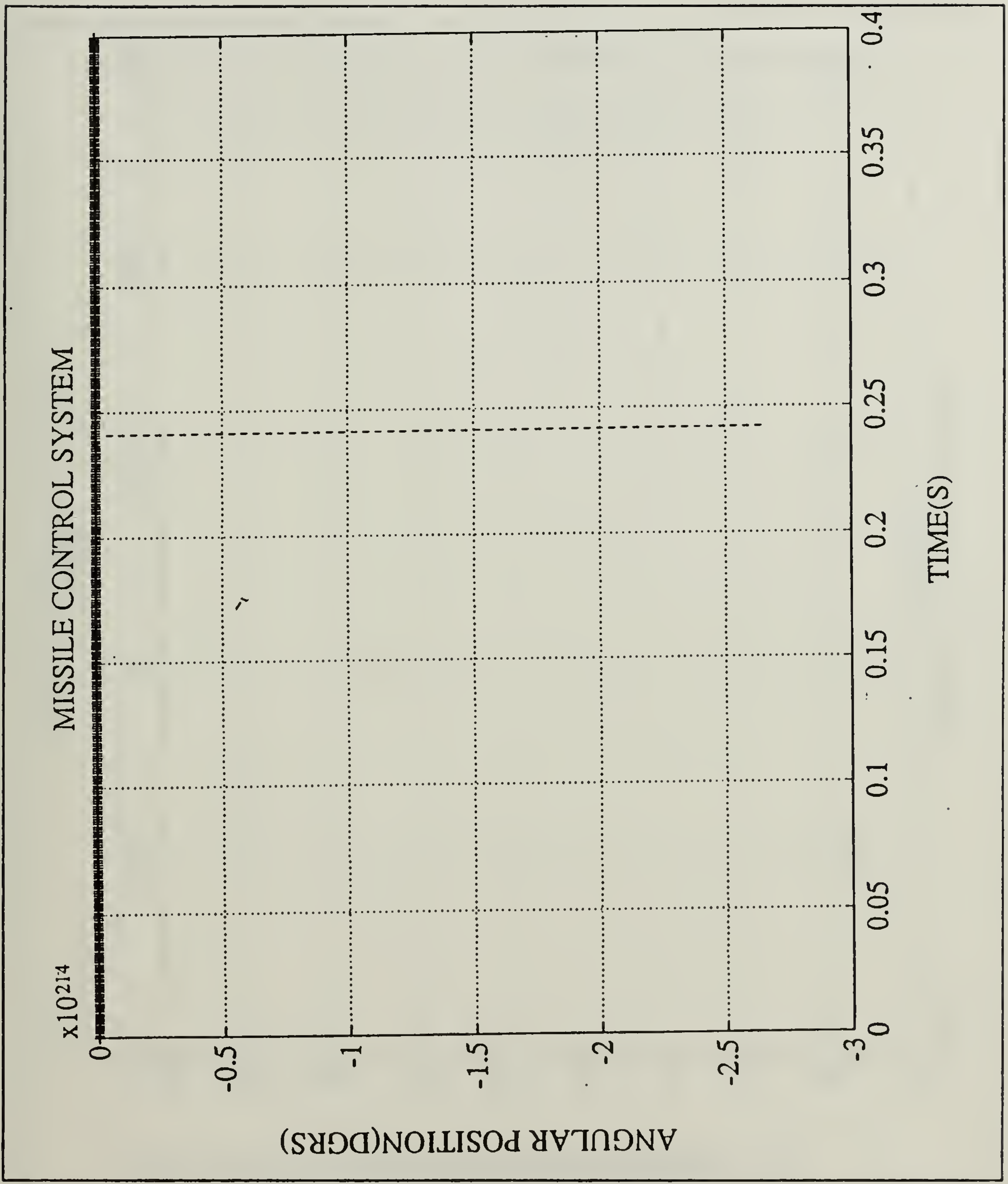

Figure 15. The desired and actual trajectory for the flexible missile with rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

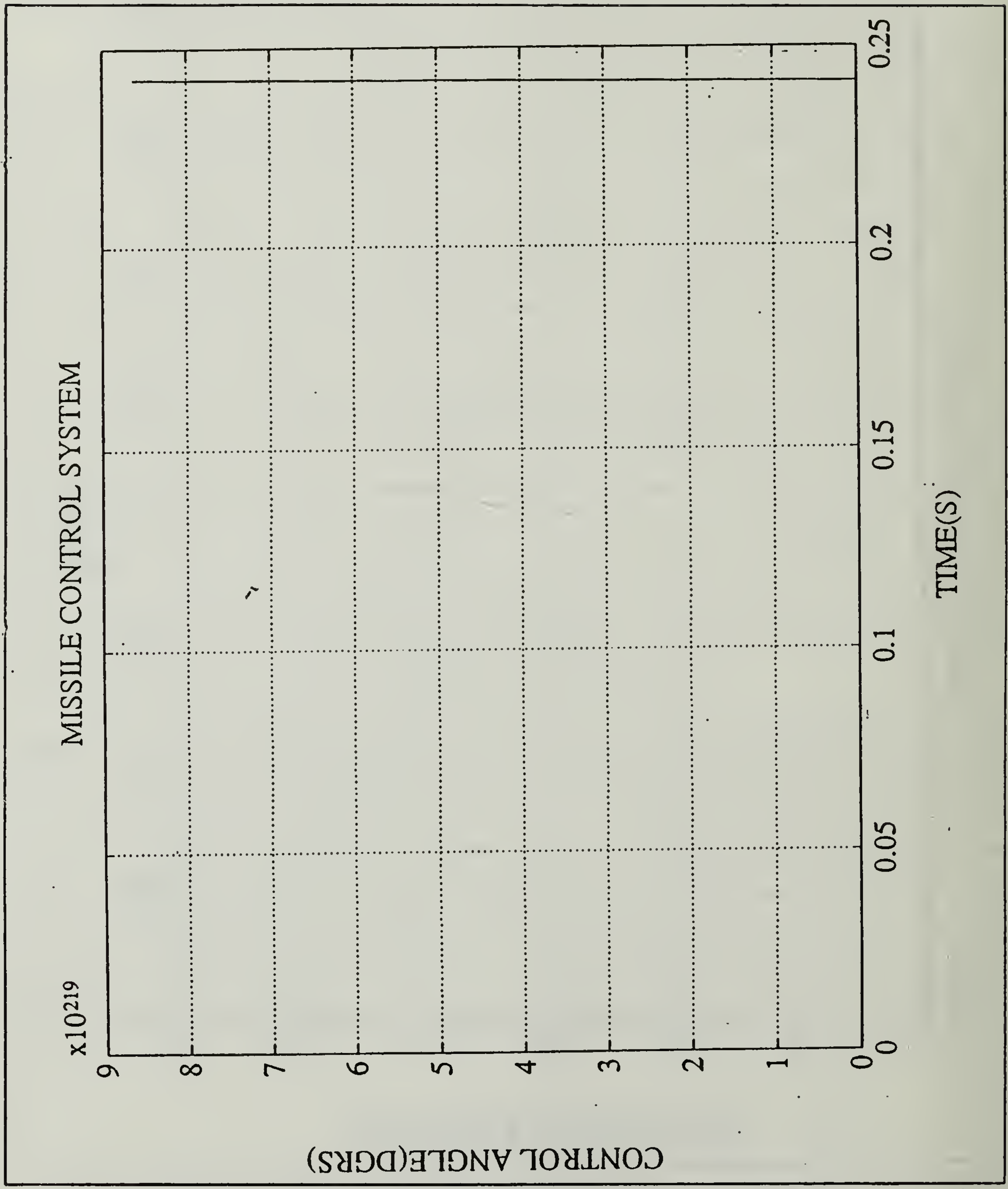

Figure 16. The control angle for the flexible missile with rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

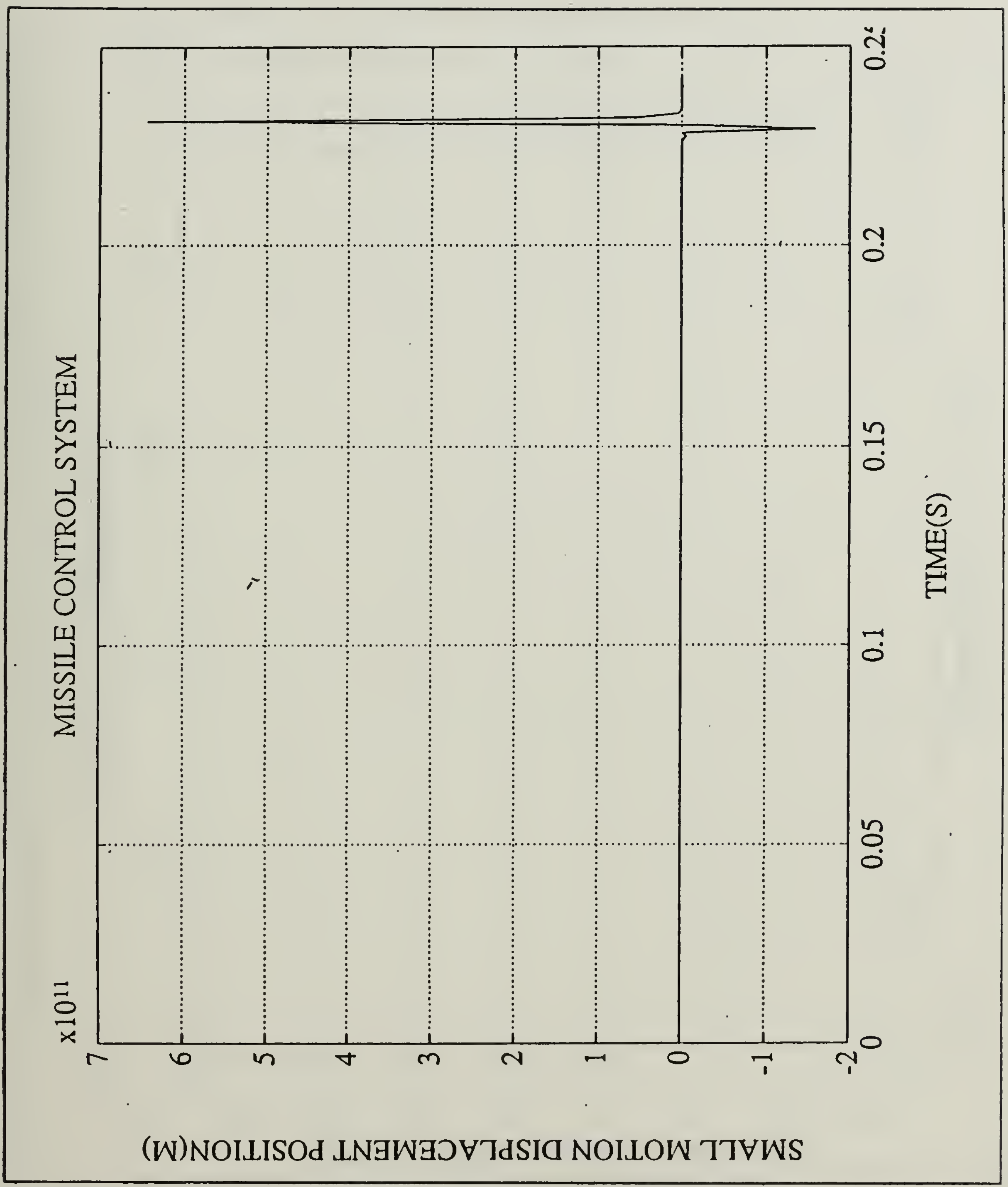

Figure 17. The small motion position v(0) for the flexible missile rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

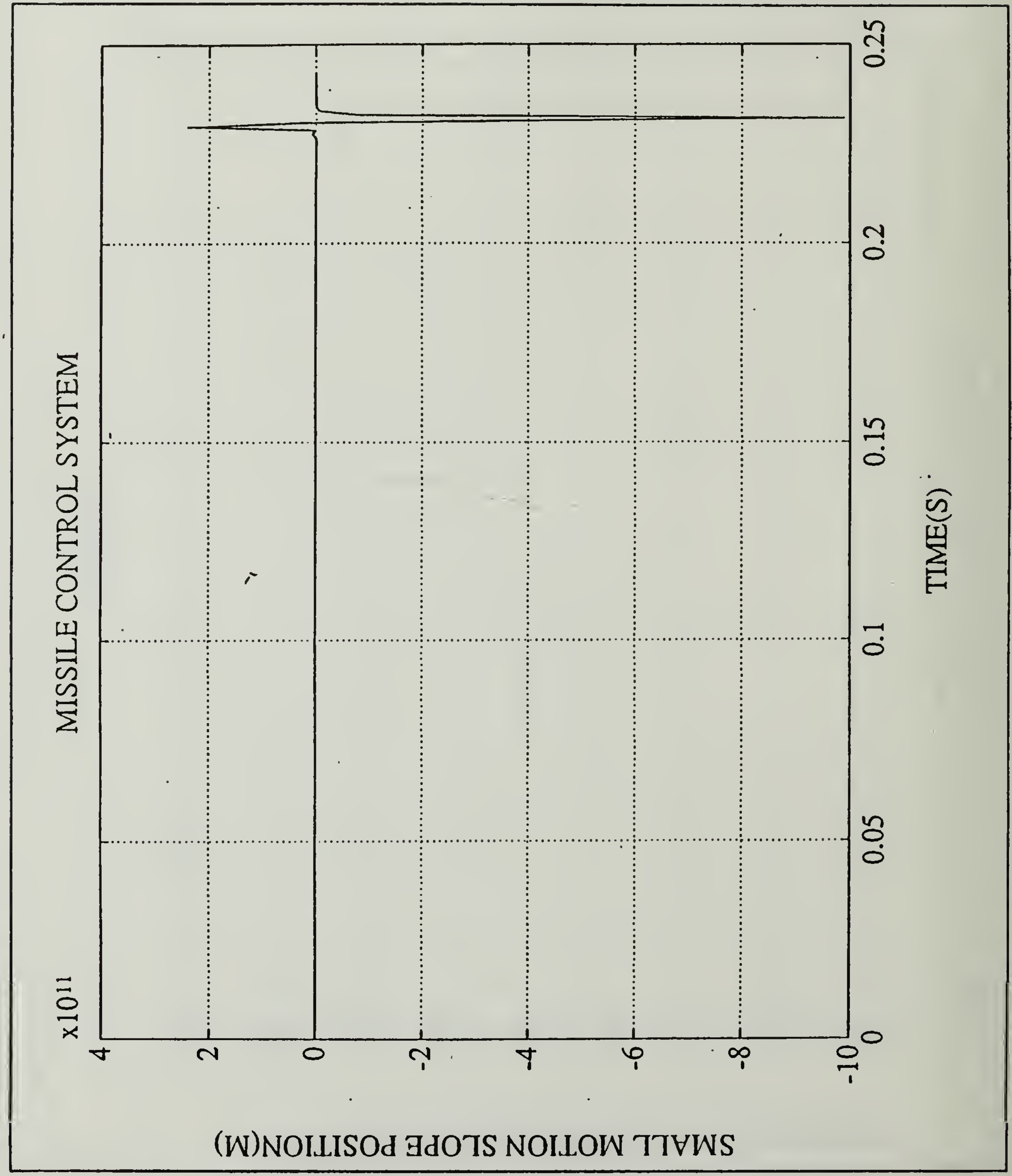

Figure 18. The small motion position phi(0) for the flexible missile rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

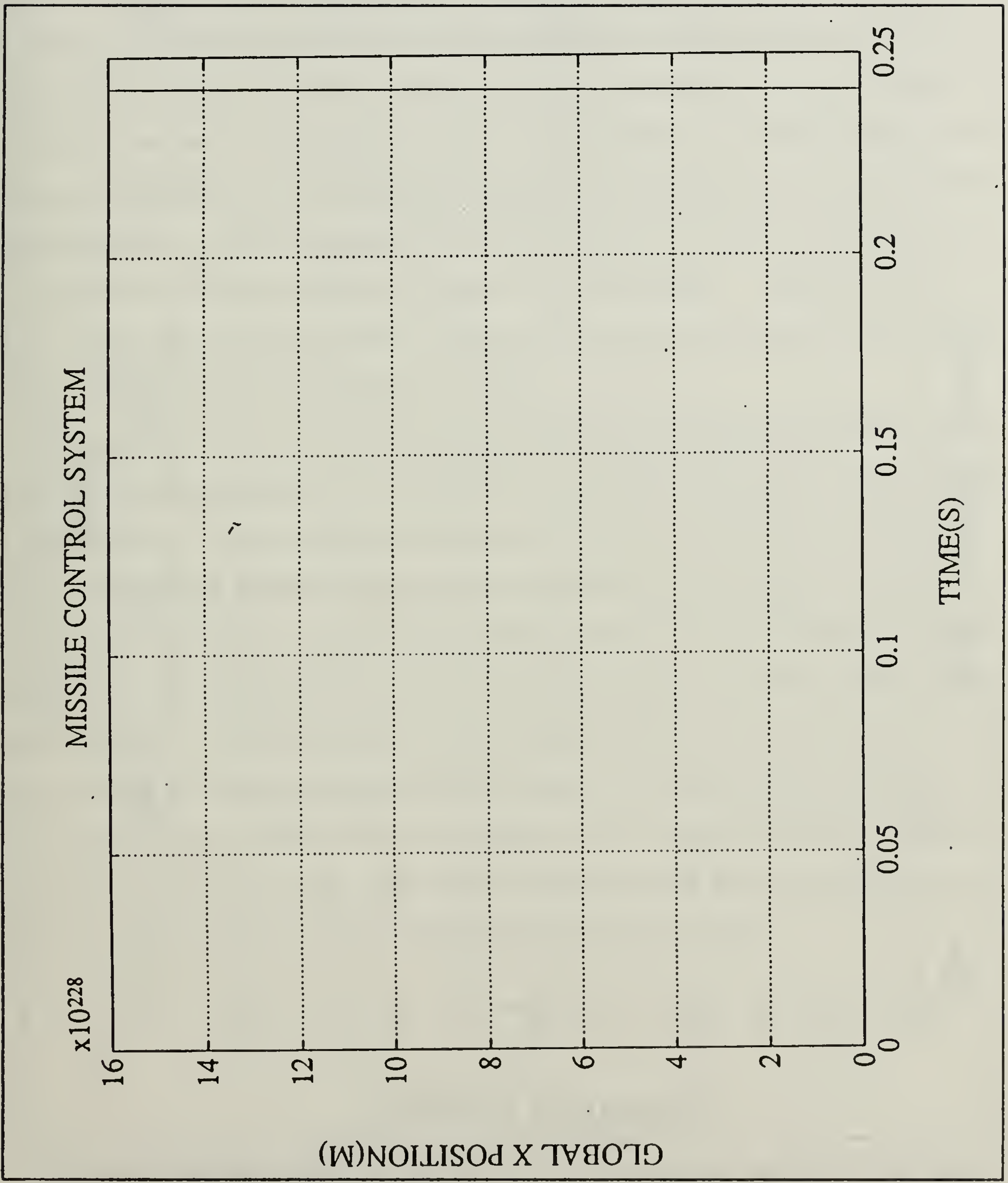

Figure 19. The large motion position X for the flexible missile rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

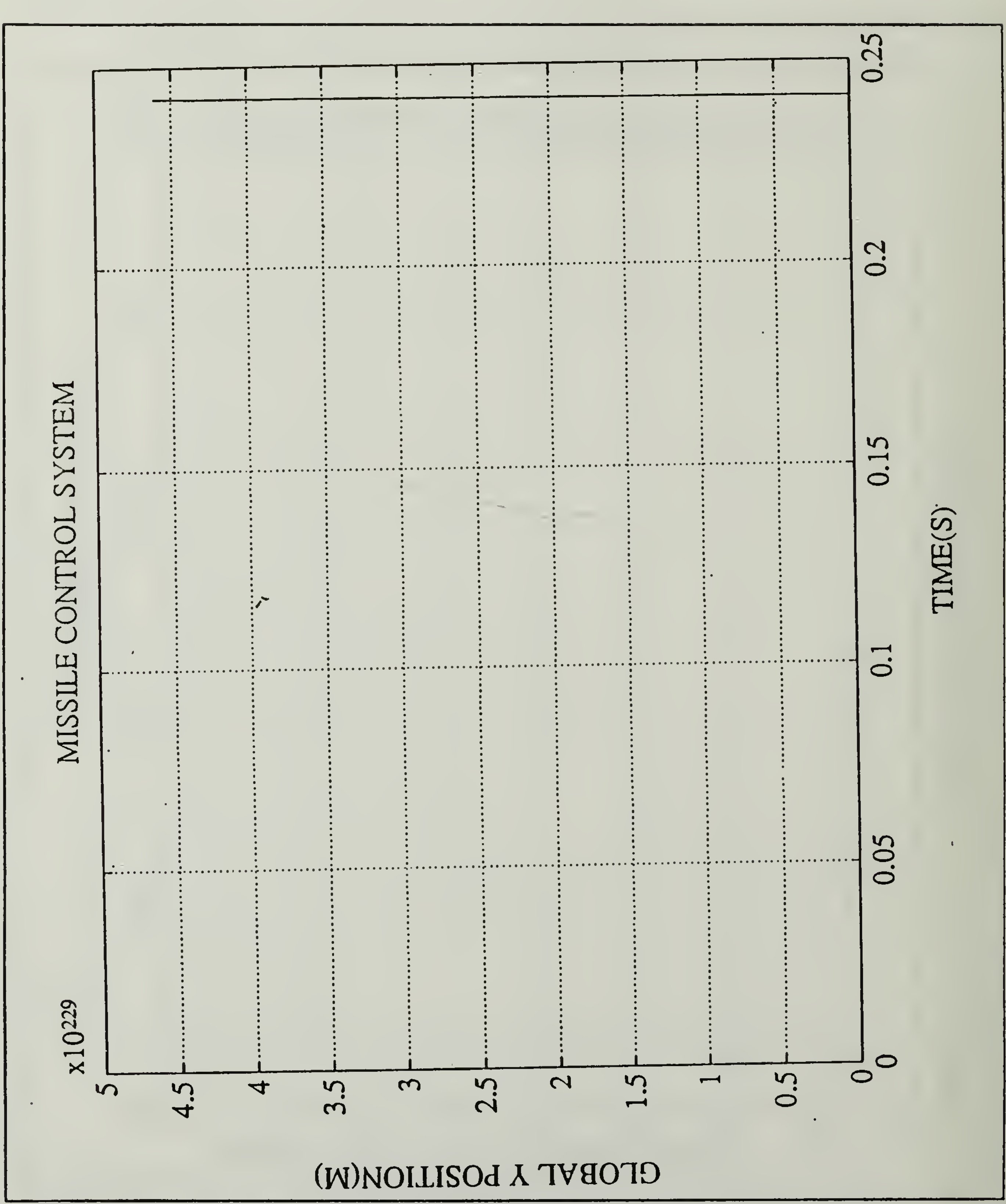

Figure 20. The large motion position Y for the flexible missile rigid-body controller without saturation on the control angle (T = 30000 N, ω_n = 300 rad/s)

Fig.21 presents the desired and actual trajectory of the flexible missile using rigid-body controller with a saturation (+/-10 Degrees) on the control angle. The force (T) and controller bandwidth (ω_n) are 30000 N and 200 rad/s, respectively. A better trajectory tracking is obtained, however the actual trajectory is oscillating about the desired trajectory which is mainly due to the switching of the control angle between the saturation lines. Fig.22 shows the control angle, and Figures 23-24 show the small motion displacement and slope of the base of the flexible missile. It was observed that the switching of the control angle between +/-10 degrees causes increased excitation of small motions. Figures 25-26 presents the position of missile base in large motion where the large motion is no longer dominated by the rigid-body motion.

The second test was to study the dynamics of the flexible missile due to an increase of the missile speed. To increase the speed of the missile, the thrust force was increased. Fig.27 presents the actual and desired trajectory of flexible missile. The force was 60000 N and controller bandwidth was 3 rad/s. As seen from Fig.27, the pitch-angle response is determined directly from the bandwidth of the controller. The increased missile speed does not change the pattern of the pitch-angle response. The control angle of the flexible missile is shown in Fig.28. The control angle is affected by the increased speed. When the speed is increased, the control angle gets smaller because a smaller control angle generates a sufficient correction to the pitch angle. The increased missile speed has little effect on small motions (Figures 29-30). Figures 31-32 show the large motions.

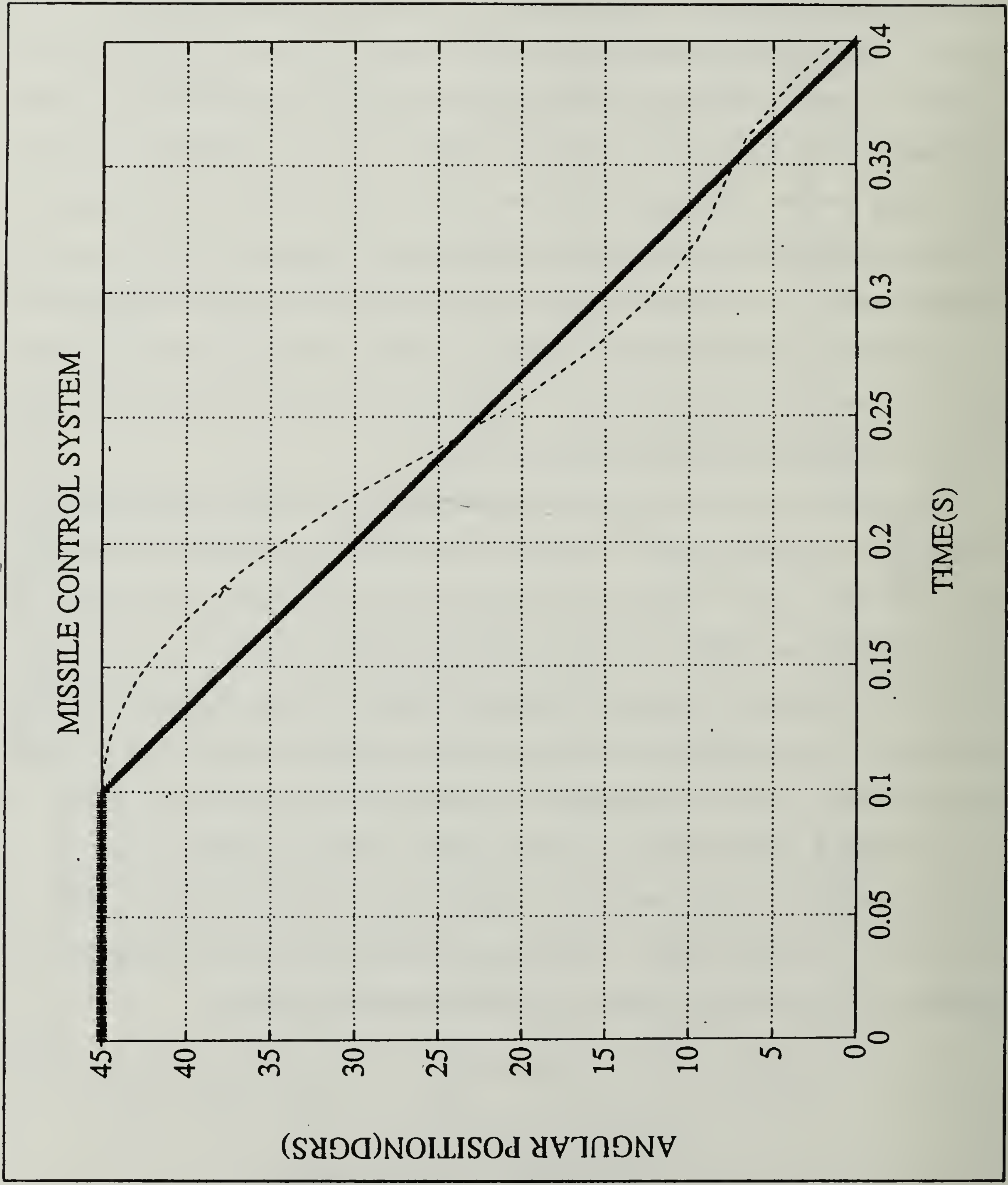

Figure 21. The desired and actual trajectory for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

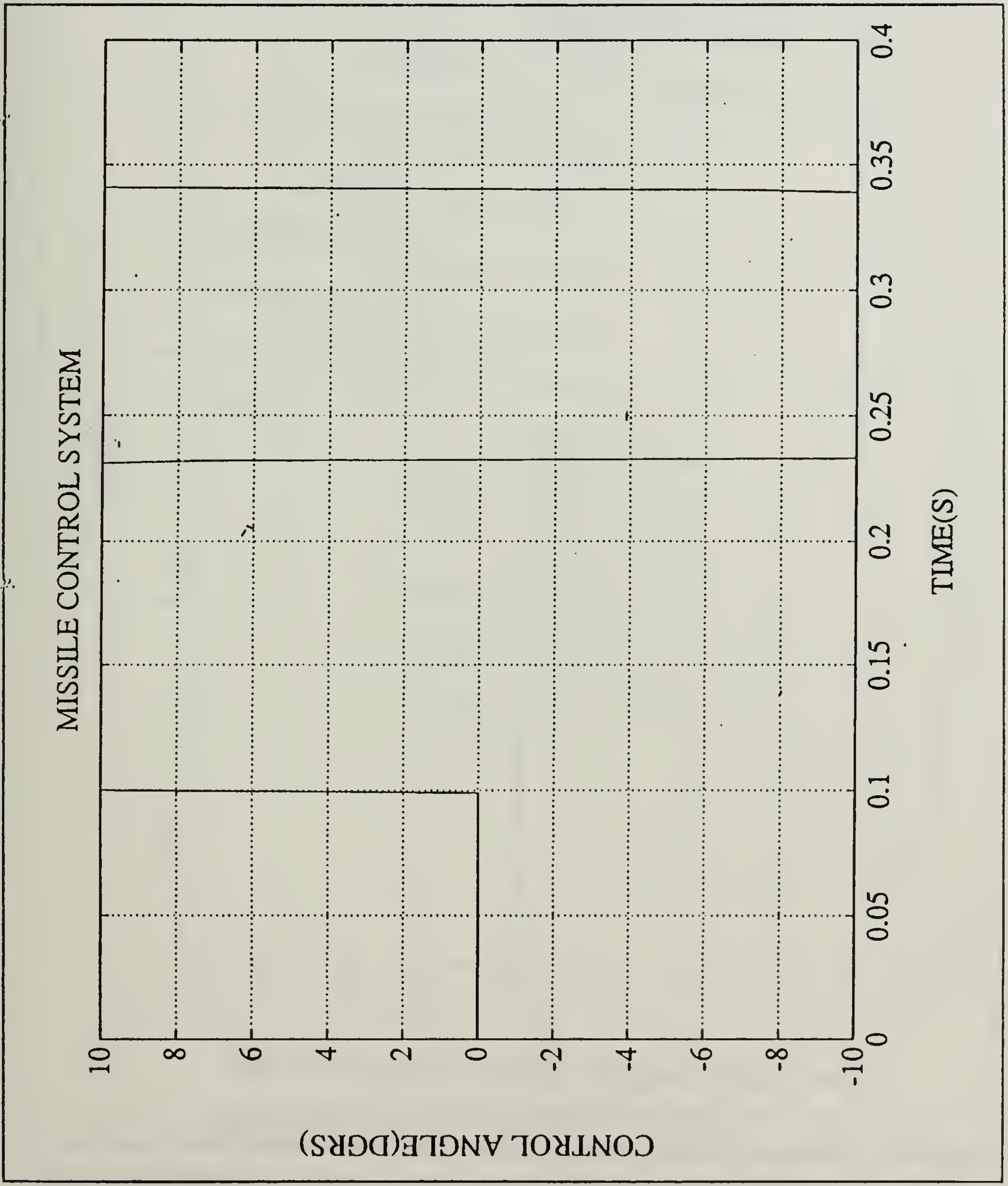

Figure 22. The control angle for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

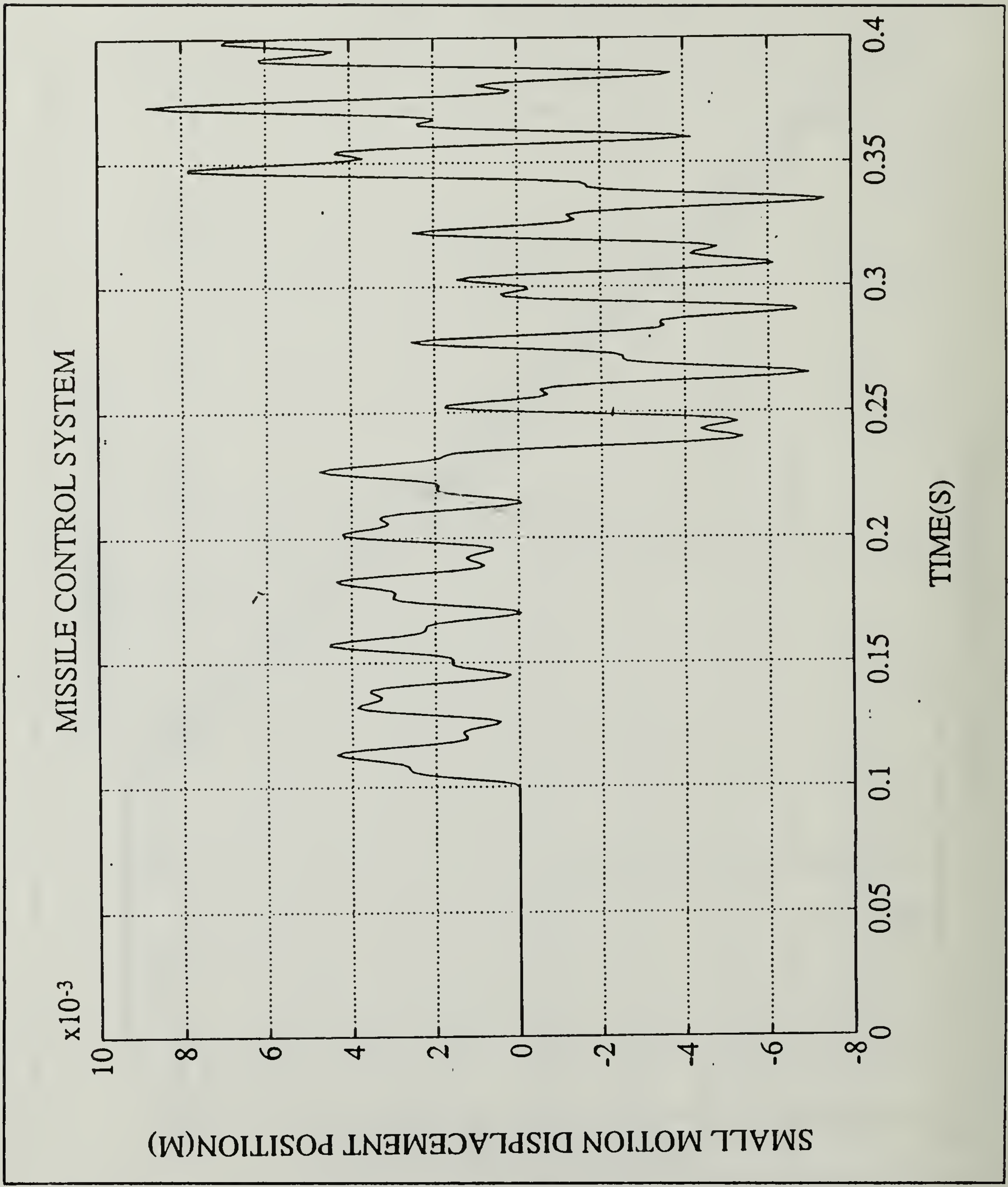

Figure 23. The small motion position v(0) for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

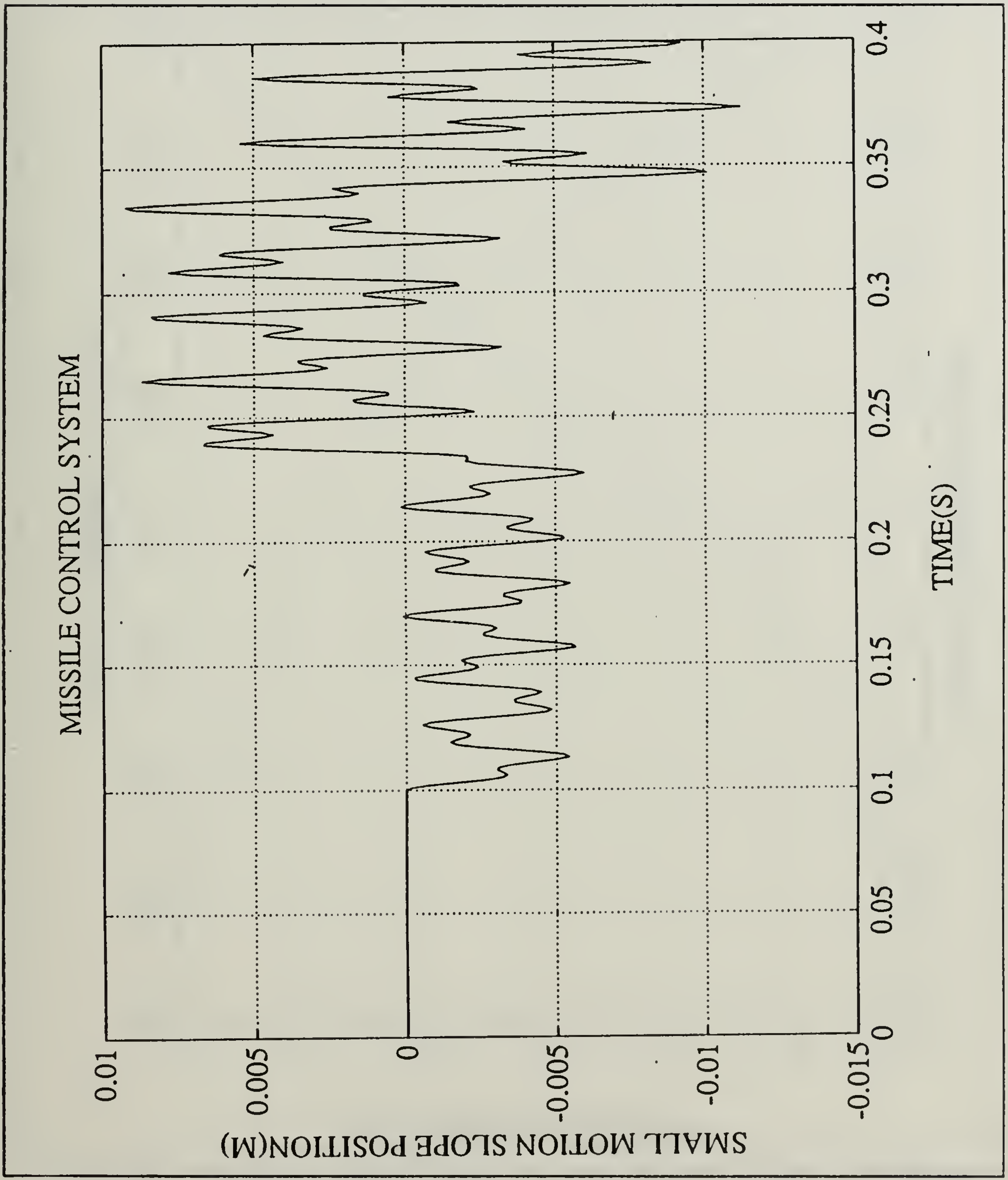

Figure 24. The small motion position phi(0) for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

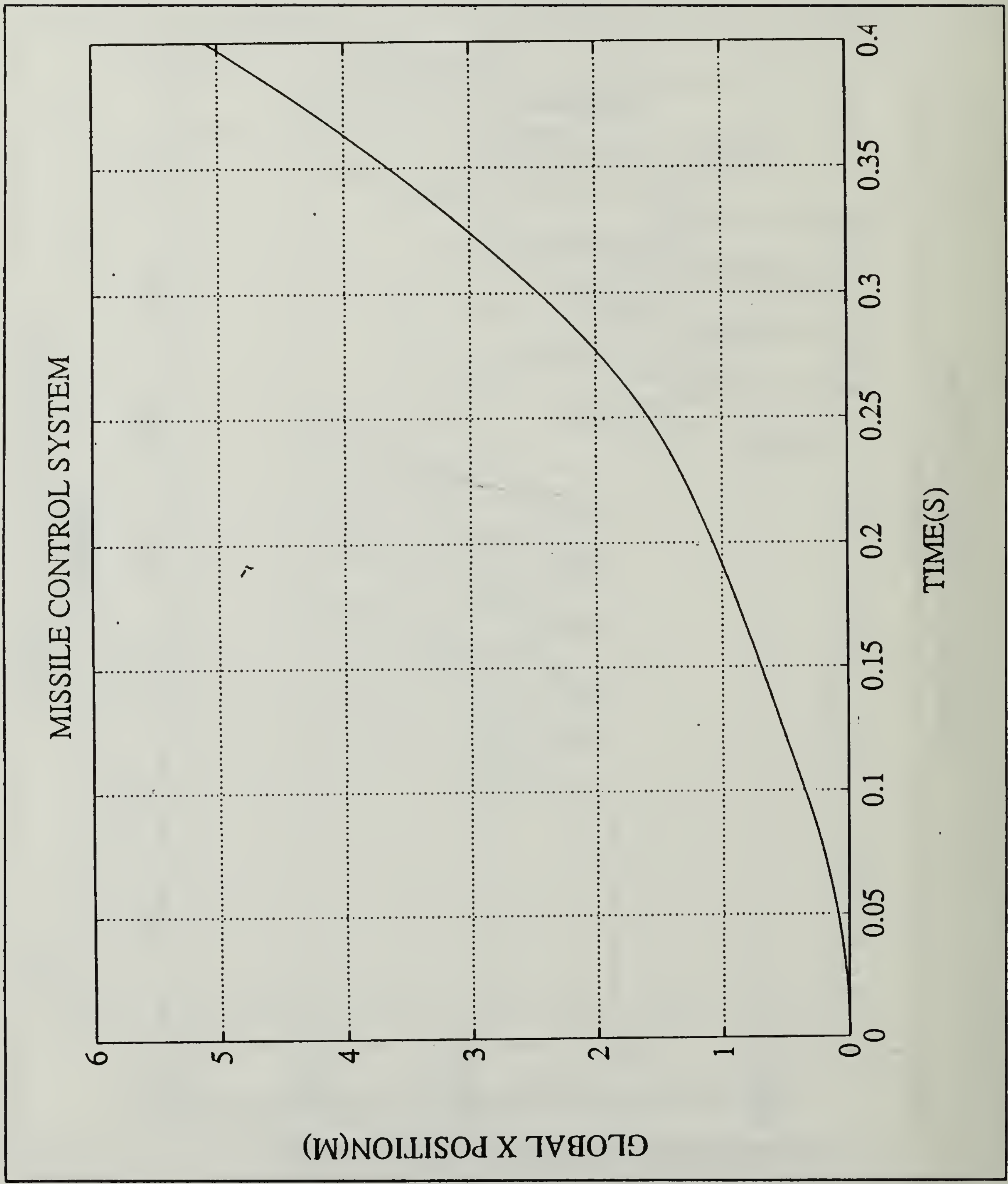

Figure 25. The large motion position X for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

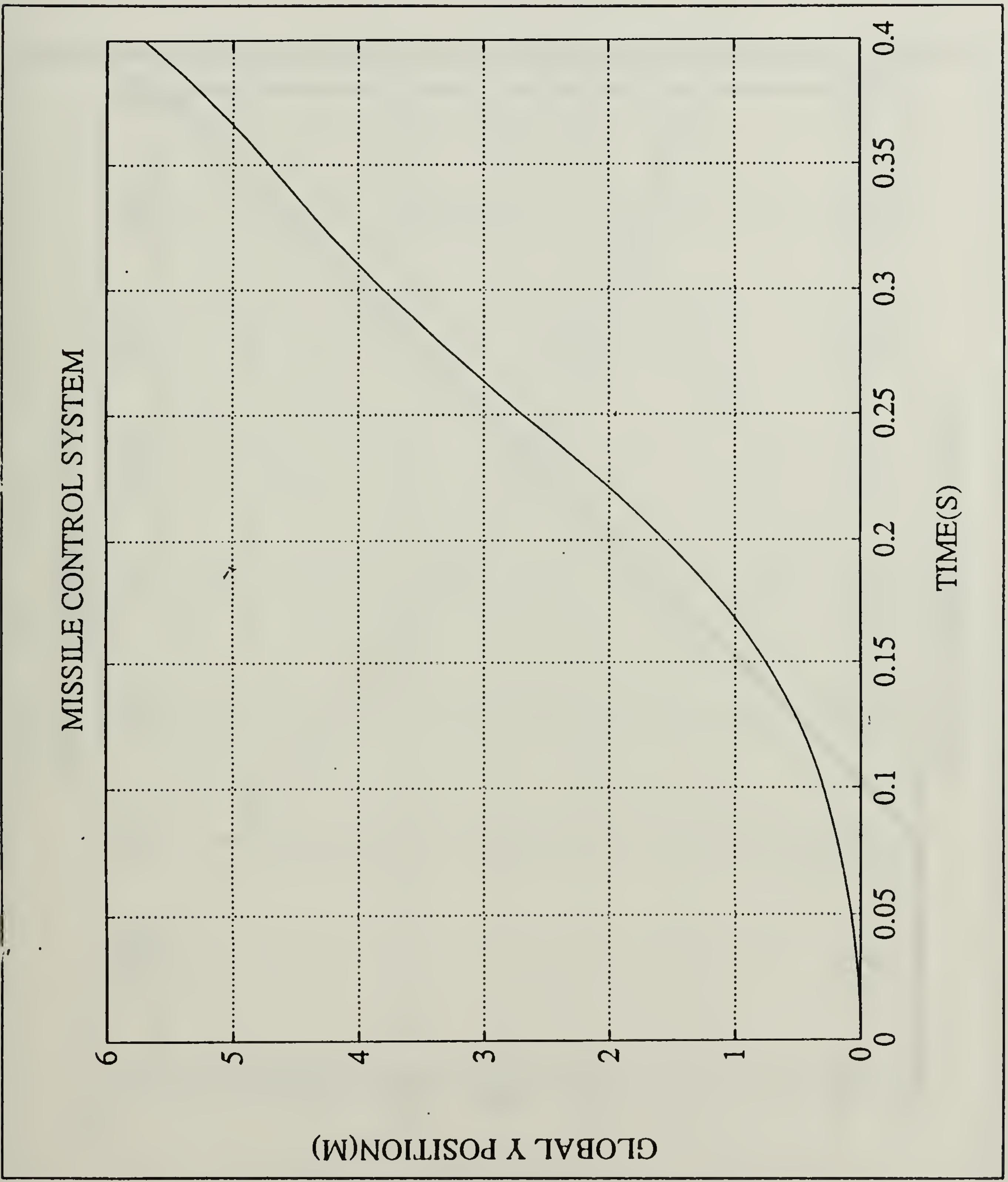

Figure 26. The large motion position Y for the flexible missile using rigid-body controller with saturation on the control angle (T = 30000 N, ω_n = 200 rad/s)

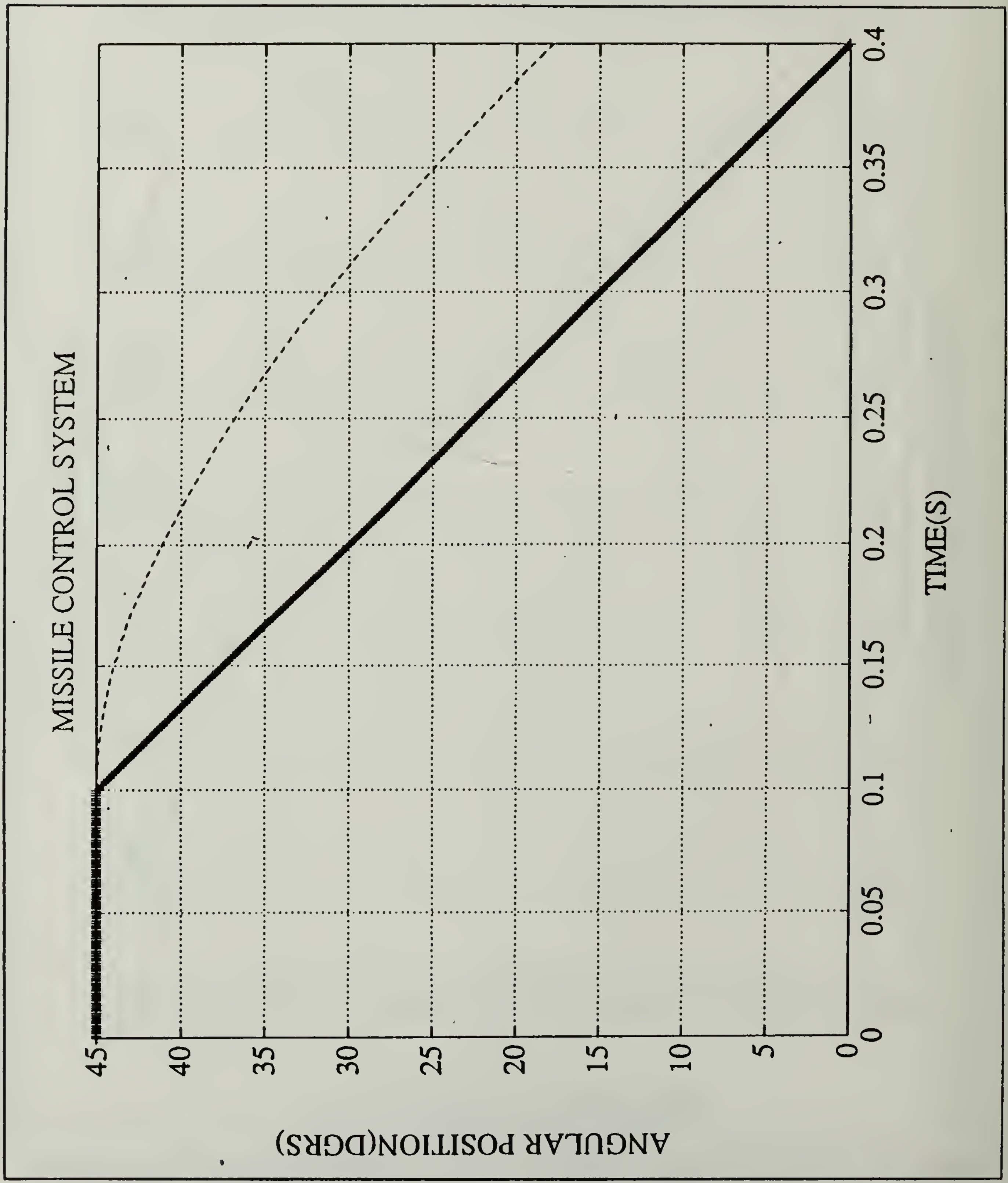

Figure 27. The desired and actual trajectory for the flexible missile using rigid-body controller with increased speed (T = 60000 N, ω_n = 3 rad/s)

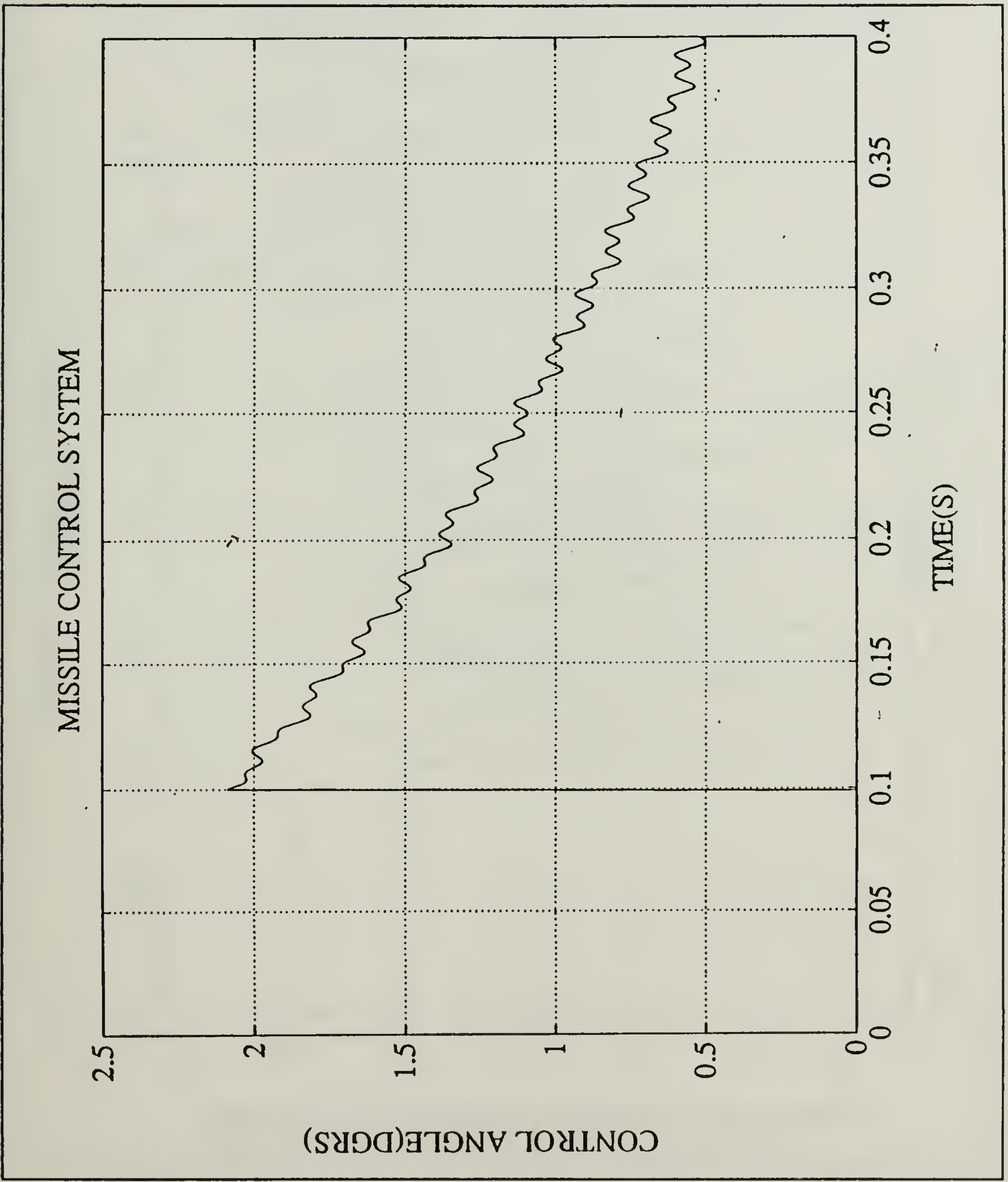

Figure 28. The control angle for the flexible missile using rigid-body controller with increased speed (T = 60000 N, ω_n = 3 rad/s)

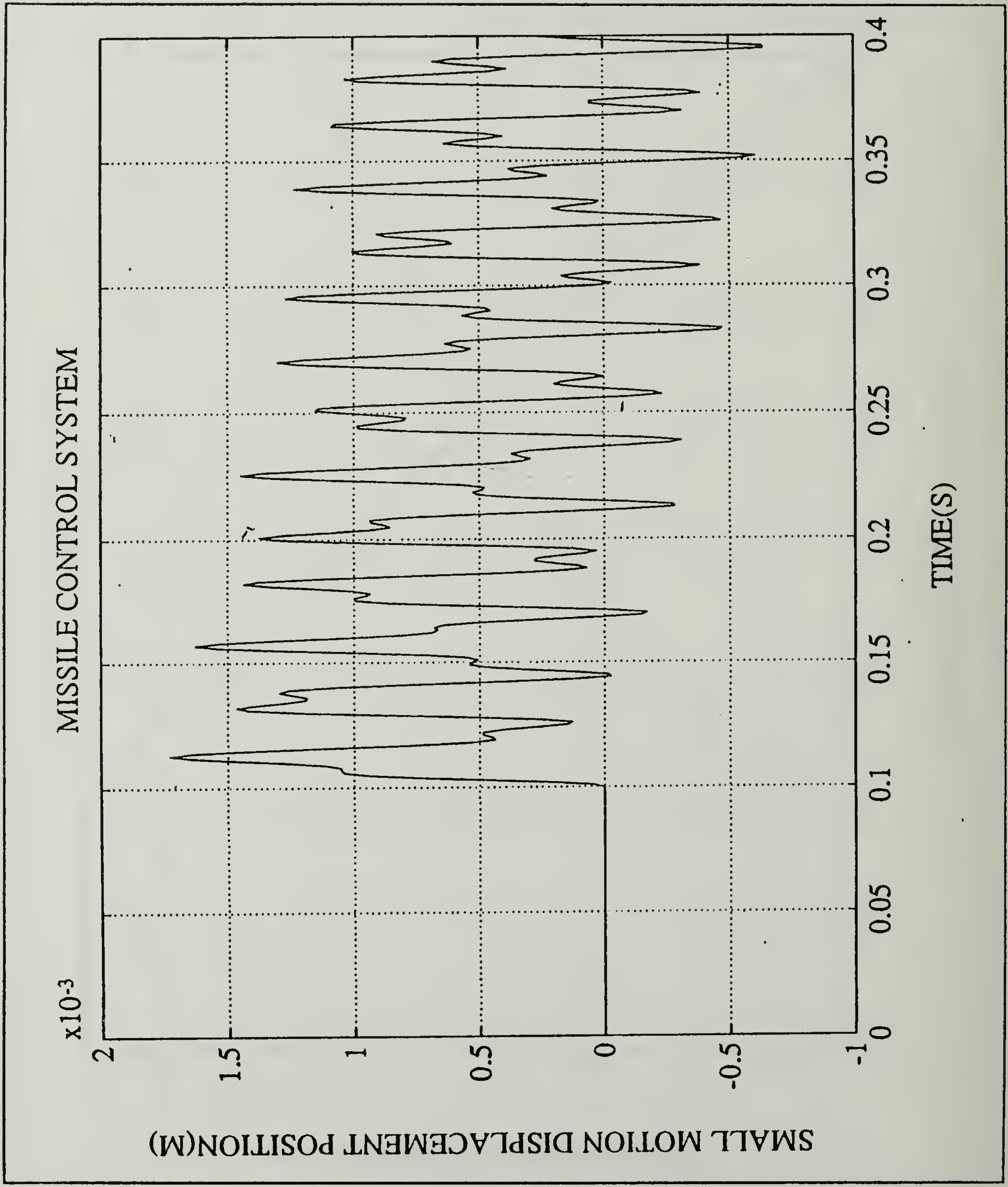

Figure 29. The small motion position v(0) for the flexible missile using rigid-body controller with increased speed. (T = 60000 N, ω_n = 3 rad/s)

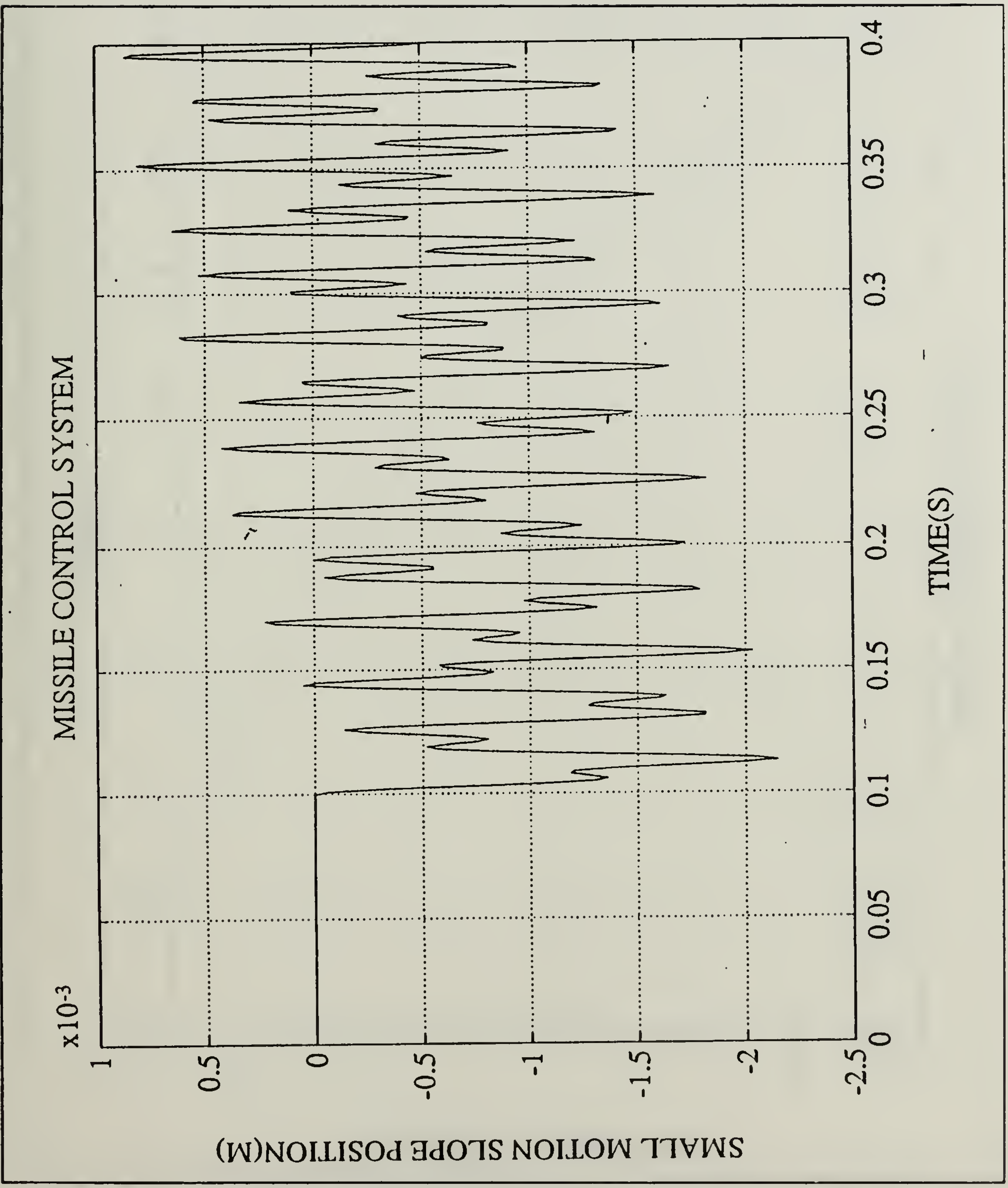

Figure 30. The small motion position phi(0) for the flexible missile using rigid-body controller with increased speed (T = 60000 N, ω_n = 3 rad/s)

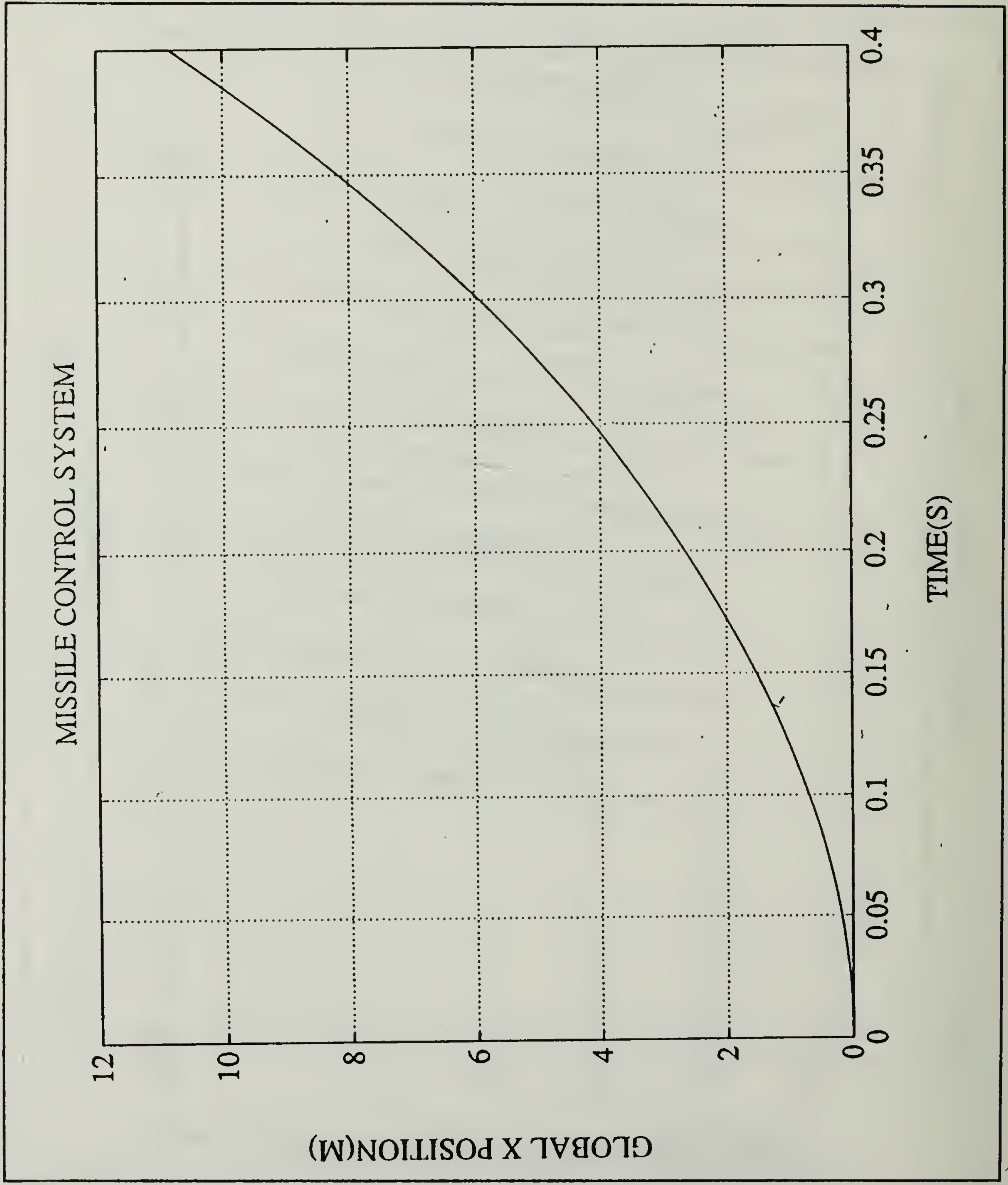

Figure 31. The large motion position X for the flexible missile using rigid-body controller with increased speed (T = 60000 N, ω_n = 3 rad/s)

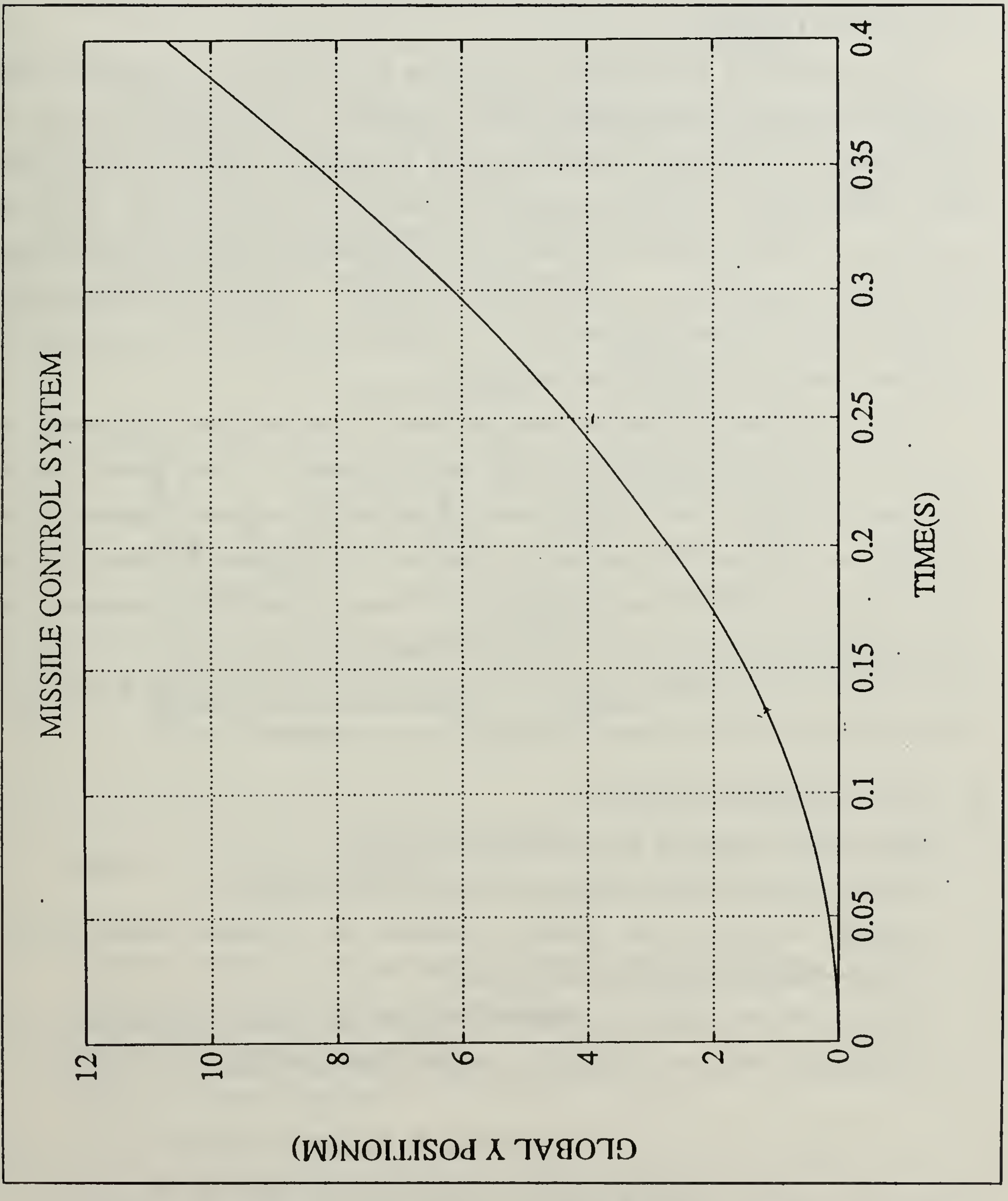

Figure 32. The large motion position Y for the flexible missile using rigid-body controller with increased speed (T = 60000 N, ω_n = 3 rad/s)

V. SUMMARY

A. CONCLUSIONS

The development and computer simulation is presented for a bending flexible missile with a rigid-body controller system moving in a 2-D coordinate frame. In this research, the dynamic model has been developed through the Equivalent Rigid Link System utilizing Lagrangian dynamics to obtain a type of system equations of motion suited for Sequential Integration Method that integrates large motion explicitly and small motion implicitly. The spatial finite element discretization of missile structure and the application of truncated natural modal responses provide an approximate solution.

The analysis and simulation were performed to understand the dynamic behavior of a flexible missile using a rigid-body controller. It was found that the controller bandwidth must be much lower than the fundamental frequency of the missile in order to use the rigid-body controller. The payload will affect the natural frequency of the missile structure i.e, when the payload is increased, the system fundamental frequency will be decreased. The payload must then be limited to achieve high-speed response. In order to increase the payload and maintain high-speed control response, a flexible-body controller is needed.

B. RECOMMENDATIONS

Areas which remain to be investigated include :

1.Add the payload and aerodynamic effects to the model.

2.Design and study the dynamic behavior of a flexible missile with a flexible-body controller in 2-D motion.

3.Build a flexible missile in a laboratory scale and obtain experimental data.

4.Design and simulate a control system for flexible missiles in 3-D motion.

APPENDIX A. DERIVATION OF THE MODE SHAPE RESPONSE MATRIX FOR THE FLEXIBLE MISSILE

In this study, the natural mode shape functions of a beam are used to represent the flexural motion of the flexible missile. Only the first two mode shapes are used. The flexible missile is modeled as a continuous Euler-Bernoulli free-free beam, neglecting shear deformation and rotary inertia effects.

The bar (Fig.33) has system parameters:

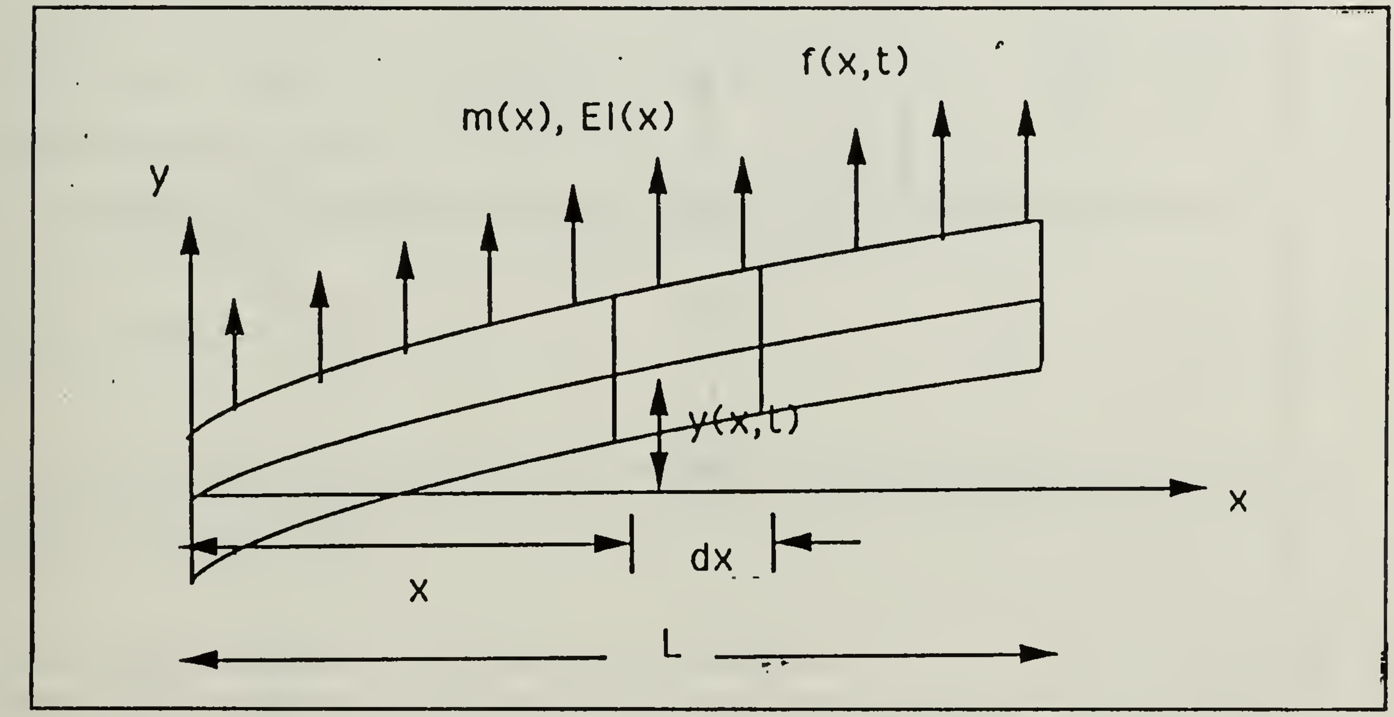

Figure 33. The Bar in Flexure

$y(x,t)$ = Transverse displacement at any point x and time t

$f(x,t)$ = Transverse force per unit length

$m(x)$ = The mass per unit length

$EI(x)$ = Flexural rigidity

E = Young's modulus of elasticity

$I(x)$ = The cross-sectional area moment of inertia

$Q(x,t)$ = Shearing force

$M(x,t)$ = Bending moment

Fig.34 shows the free body diagram corresponding to a bar element of length dx.

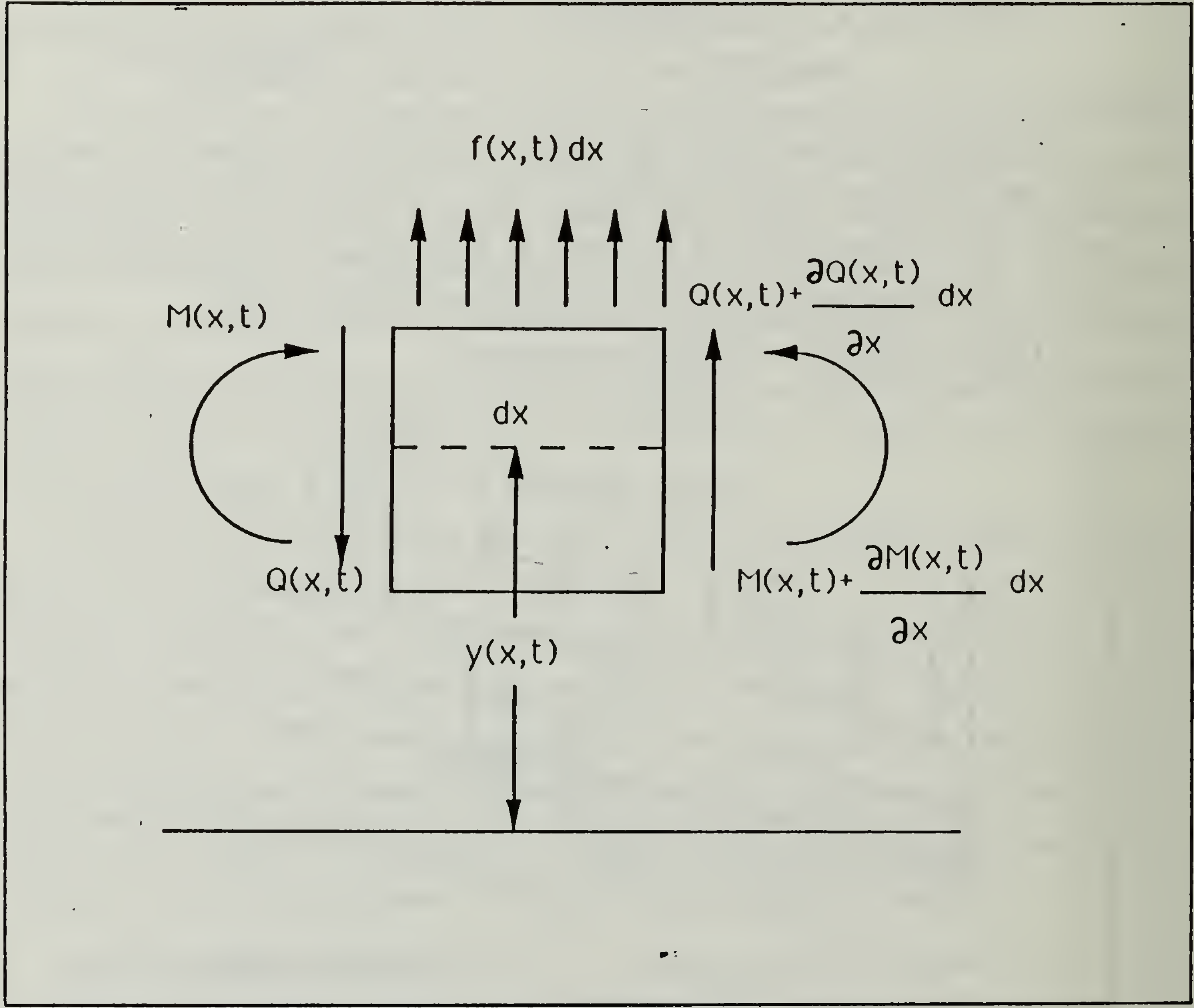

Figure 34. Free Body Diagram Corresponding to a Bar Element

The force balance of the free body is

$$[Q(x,t) + \frac{\partial Q(x,t)}{\partial x} dx] - Q(x,t) + f(x,t)dx = m(x)dx \frac{\partial^2 y(x,t)}{\partial t^2} \qquad (A-1)$$

The moment equation of motion about the axis normal to x and y, ignoring the inertia torque associated with rotation,

$$[M(x,t)+\frac{\partial M(x,t)}{\partial x}dx]-M(x,t)+[Q(x,t)+\frac{\partial Q(x,t)}{\partial x}dx]dx$$

$$+f(x,t)dx\,\frac{dx}{2}=0 \qquad (A-2)$$

Canceling appropriate terms, ignoring terms involving second order terms in dx and combining Eqs. (A-1) and (A-2),

$$-\frac{\partial^2 M(x,t)}{\partial x^2}+f(x,t)=M(x,t)\frac{\partial^2 y(x,t)}{\partial t^2} \qquad (A-3)$$

Eq. (A-3) relates the bending moment M(x,t), transverse force f(x,t) and bending displacement y(x,t).

The relation between the bending moment and the bending deformation is

$$M(x,t)=EI(x,t)\frac{\partial^2 y(x,t)}{\partial x^2} \qquad (A-4)$$

Inserting Eq. (A-3) into Eq. (A-4), we obtain the differential equation for the flexural vibration of a bar,

$$-\frac{\partial^2}{\partial x^2}(EI(x)\frac{\partial^2 y(x,t)}{\partial x^2})+f(x,t)=M(x,t)\frac{\partial^2 y(x,t)}{\partial t^2} \qquad (A-5)$$

The bending moment and shearing force of the free ends (x=0, x=L) are zero.

$$EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}\Big|_{x=0}=0 \qquad EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}\Big|_{x=L}=0 \qquad (A-6)$$

$$\frac{\partial}{\partial x}[EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}]\Big|_{x=0}=0 \qquad \frac{\partial}{\partial x}[EI(x)\frac{\partial^2 y(x,t)}{\partial x^2}]\Big|_{x=L}=0 \qquad (A-7)$$

Eqs. (A-6) and (A-7) are called natural boundary conditions.

Considering the free vibration characterized by f(x,t) = 0, using separation of variables method and simplifying Eq. (A-5),

$$\frac{d^2}{dx^2}[EI(x)\frac{d^2Y(x)}{dx^2}] = w^2 m(x)Y(x) \qquad (A-8)$$

Simplifying the Eq. (A-8), for EI(x) = constant

$$\frac{d^4Y(x)}{dx^4} - \beta^4 Y(x) = 0 \qquad (A-9)$$

where $\beta^4 = \dfrac{w^2 m(x)}{EI(x)}$

The boundary conditions require that,

At x = 0 (base)

$$\frac{d^2Y(x)}{dx^2}|_{x=0} = 0$$

$$\frac{d^3Y(x)}{dx^3}|_{x=0} = 0 \qquad (A-10)$$

At x = L (tip)

$$\frac{d^2Y(x)}{dx^2}|_{x=L} = 0$$

$$\frac{d^3Y(x)}{dx^3}|_{x=L} = 0 \qquad (A-11)$$

The general solution of Eq. (A-9),

$$Y(x) = C_1\sin(\beta x) + C_2\cos(\beta x) + C_3\sinh(\beta x) + C_4\cosh(\beta x) \qquad (A-12)$$

Taking the derivatives of Eq. (A-12), and substuting boundary conditions,

$$Y(x) = C_1(\sin\beta x + \sinh\beta x) + C_2(\cos\beta x + \cosh\beta x) \qquad (A-13)$$

Solving Eq. (A-13) yields,

$$\cos\beta L \cosh\beta L = 1 \qquad (A-14)$$

The first five consecutive roots of this equation are;

$\beta_1 = 0.0$

$\beta_2 = 4.712388$

$\beta_3 = 7.853981$

$\beta_4 = 10.995574$

$\beta_5 = 14.137166$

Eq. (A-13) is now written in the following form,

$$Y_r(x) = C_r(\cos\beta_r x + \cosh\beta_r x) + (\sin\beta_r x + \sinh\beta_r x) \quad r = 1,2 \qquad (A-15)$$

where:

$$C_r = \frac{\sin\beta_r L - \sinh\beta_r L}{-\cos\beta_r L + \cosh\beta_r L} \qquad (A-16)$$

The transverse displacement v(x) and slope $\Phi(x)$ can be represented in the following forms respectively,

$$v(x) = \sum_{r=1}^{2} a_r Y_r(x) \qquad (A-17)$$

$$\Phi(x) = \frac{\partial v}{\partial x} \qquad (A-18)$$

$$v(x) = a_1(C_1(\cos\beta_1 x + \cosh\beta_1 x) + (\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ a_2(C_2(\cos\beta_2 x + \cosh\beta_2 x) + (\sin\beta_2 x + \sinh\beta_2 x)) \qquad (A-19)$$

$$\Phi(x) = a_1(C_1\beta_1 x + \sinh\beta_1 x) + \beta_1(\cos\beta_1 x + \cosh\beta_1 x))$$

$$+ a_2(C_2\beta_2(-\sin\beta_2 x + \sinh\beta_2 x) + (\cos\beta_2 x + \cosh\beta_2 x)) \qquad (A-20)$$

Substituting the boundary conditions into shape function gives,

$$v(0) = 2a_1C_1 + 2a_2C_2 \tag{$A-21$}$$

$$\Phi(0) = 2a_1\beta_1 + 2a_2\beta_2 \tag{$A-22$}$$

The modal amplitudes a_1 and a_2;

$$a_1 = (\frac{1}{2C_1} + \frac{C_2\beta_1}{C_1^2E})v(0) + \frac{C_2}{c_1E}\,\Phi(0) \tag{$A-23$}$$

$$a_2 = -\frac{\beta_1}{C_1E}\,v(0) + \frac{1}{E}\,\Phi(0) \tag{$A-24$}$$

$$v = av(0) + b\Phi(0) \tag{$A-25$}$$

$$a = F_1Y_1(x) + F_3Y_2(x) \tag{$A-26$}$$

$$b = F_2Y_1(x) + F_4Y_2(x) \tag{$A-27$}$$

$$F_1 = \frac{1}{2C_1} + \frac{C_2\beta_1}{C_1^2E} \tag{$A-28$}$$

$$F_2 = \frac{-C_2}{C_{1E}} \tag{$A-29$}$$

$$F_3 = -\frac{\beta_1}{C_1E} \tag{$A-30$}$$

$$F_4 = \frac{1}{E} \tag{$A-31$}$$

$$E = 2\beta_2 - 2\frac{C_2\beta_1}{C_1} \tag{$A-32$}$$

Substituting Eqs. (A-15), (A-16), (A-26), (A-27), (A-28), (A-29), (A-30), (A-31), (A-32) into Eq. (A-25) yields an expression for the transverse displacement of a flexible missile as a function of the missile base nodal displacements, v(0) and Φ(0),

$$v(x,t) = [F_1(C_1(\cos\beta_1 x + \cosh\beta_1 x) + (\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_3(C_2(\cos\beta_2 x + \cosh\beta_2 x) + (\sin\beta_2 x + \sinh\beta_2 x))]v(0)$$

$$+ [F_2(C_1(\cos\beta_1 x + \cosh\beta_1 x) + (\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_4(C_2(\cos\beta_2 x + \cosh\beta_2 x) + (\sin\beta_2 x + \sinh\beta_2 x))]\Phi(0) \qquad (A-33)$$

This expression is differentiated twice to obtain $v''(x)$, which is necessary for the calculation of the potential energy due to deformation and theoretical strain,

$$v''(x,t) = (F_1(C_1\beta_1^2(-\cos\beta_1 x + \cosh\beta_1) + \beta_1^2(-\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_3(C_2\beta_2^2(-\cos\beta_2 x + \cosh\beta_2) + \beta_2^2(-\sin\beta_2 x + \sinh\beta_2 x)))v(0)$$

$$+ (F_2(C_1\beta_1^2(-\cos\beta_1 x + \cosh\beta_1) + \beta_1^2(-\sin\beta_1 x + \sinh\beta_1 x))$$

$$+ F_4(C_2\beta_2^2(-\cos\beta_2 x + \cosh\beta_2) + \beta_2^2(-\sin\beta_2 x + \sinh\beta_2 x)))\Phi(0) \qquad (A-35)$$

Now substitution of v(x) into the 3x1 deformation vector, $\vec{d}$, yields the 3x2 shape function matrix, ϕ, and the 2x1 nodal displacement vector, $\vec{U}$,

$$d = \phi v \Rightarrow \begin{bmatrix} 0 \\ 0 \\ v(x) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ a & b \end{bmatrix} \begin{bmatrix} v(0) \\ \Phi(0) \end{bmatrix} \qquad (A-36)$$

The shape function matrix is now in a form convenient for computer coding.

APPENDIX B. DERIVATION OF THE EQUATIONS OF MOTION FOR THE FLEXIBLE MISSILE

The ERLS is used to separate the flexible missile's motion into a small motion and a large motion. The large motion generalized coordinates are defined as X_0, Y_0, theta (θ)and the small motion generalized coordinates as U. The Lagrangian Dynamics are very helpful in the derivation of the equations of motion of complex systems. U represents a column vector which has two elements. First one is v(0) which is the nodal displacement of the flexible missile's base and Φ(0) which is the slope of the flexible missile's base. We can represent the vector of the generalized coordinates by Eq. (B-1),

$$\vec{\xi} = [\ X_0 \quad Y_0 \quad \theta\ ]^T \qquad \vec{U} = [v(0) \quad \Phi(0)\ ]^T \qquad (B-1)$$

These generalized coordinates will be used in the application of the Lagrange equations to the equations of motion for both small and large motion coordinate stems. The Lagrangian equations are defined in Eqs. (B-2) and (B-3).

$$\frac{d}{dt}[\frac{\partial KE}{\partial \dot{\xi}_i}] - \frac{\partial KE}{\partial \xi_i} + \frac{\partial PE}{\partial \xi_i} = F_{\xi i} \quad (i = 1,2,3) \qquad (B-2)$$

$$\frac{d}{dt}[\frac{\partial KE}{\partial \dot{\vec{U}}}] - \frac{\partial KE}{\partial \vec{U}} + \frac{\partial PE}{\partial \vec{U}} = F_u \qquad (B-3)$$

where ξ_i is a component of ξ

Using ERLS we can define the global position of the base position in terms of the local position vector ($\vec{r}$), a deformation vector ($\vec{d}$), and a coordinate transformation matrix (W) with Eq. (B-4),

$$\vec{R} = W(\vec{r} + \vec{d}) \qquad (B-4)$$

Continuing in the development of the equations of motions, we determine the derivative with respect to time for global positions in order to obtain the kinetic energy expressions.

$$\dot{\vec{R}} = \dot{W}(\vec{r} + \vec{d}) + W\dot{\vec{d}} \qquad (B-5)$$

The kinetic and potential energy expresions follow. In this development we have seperated the large and small motion energy contributions and will present them seperately.

$$KE = \frac{1}{2}\int \dot{\vec{R}}^T \dot{\vec{R}} dm \qquad (B-6)$$

$$PE = \frac{1}{2}\int \vec{U}^T \Gamma^T C \Gamma \vec{U} dx - \int \vec{R}^T \vec{g} dm \qquad (B-7)$$

Redefining $\vec{d}$ in terms of the shape matrix ϕ (derivation of $\vec{d}$ is presented in Appendix A), we can rewrite Eq. (B-6) as

$$KE = \frac{1}{2}\int ([\dot{W}(\vec{r} + \vec{d}) + W\dot{\vec{d}}]^T [\dot{W}(\vec{r} + \vec{d}) + W\dot{\vec{d}}])dm \qquad (B-8)$$

$$KE = \frac{1}{2}\int (\vec{r}^T \dot{W}^T \dot{W} \vec{r} + 2\vec{r}^T \dot{W}^T \dot{W} \phi \vec{U} + 2\vec{r}^T \dot{W}^T W \phi \dot{\vec{U}}$$

$$+ \vec{U}^T \phi^T \dot{W}^T \dot{W} \phi \vec{U} + 2U^T \phi^T \dot{W}^T W \phi \dot{\vec{U}} + \dot{\vec{U}}^T \phi^T W^T W \phi \vec{U}^T)dm \qquad (B-9)$$

Continuing in the development of the equations of motion, we first express the derivative of the kinetic energy with respect to the time rate of change of the large motion coordinate $\dot{\xi}$, Eq. (B-13), and then determine the time rate of change of this expression, Eq. (B-14). Since

$$\vec{\xi} = \begin{bmatrix} X_0 \\ Y_0 \\ \theta \end{bmatrix} = \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{bmatrix} \qquad (B-10)$$

$$W_{,1} = \frac{\partial W}{\partial X} = \frac{\partial W}{\partial \xi_1} \equiv \text{Partial derivative w.r.t. X}$$

$$W_{,2} = \frac{\partial W}{\partial Y} = \frac{\partial W}{\partial \xi_2} \equiv \text{Partial derivative w.r.t. Y}$$

$$W_{,3} = \frac{\partial W}{\partial \theta} = \frac{\partial W}{\partial \xi_3} \equiv \text{Partial derivative w.r.t. } \theta$$

$W_{,i} \equiv$ Partial derivative of W w.r.t. X, Y, θ i=1,2,3

$$\frac{\partial KE}{\partial \dot{\xi}_i} = \frac{1}{2} \int (\vec{r}^T \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} \dot{W} \vec{r} + \vec{r}^T \dot{W}^T \frac{\partial \dot{W}}{\partial \dot{\xi}_i} \vec{r} + 2 \vec{r}^T \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} \dot{W} \phi \vec{U}$$

$$+ 2 \vec{r}^T \dot{W}^T \frac{\partial \dot{W}}{\partial \dot{\xi}_i} \phi \vec{U} + 2 \vec{r}^T \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} W \phi \vec{U} + \vec{U}^T \phi^T \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} \dot{W} \phi \vec{U}$$

$$+ \vec{U}^T \phi^T \dot{W}^T \frac{\partial \dot{W}}{\partial \dot{\xi}_i} \phi \vec{U} + 2 \vec{U}^T \phi^T \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} W \phi \vec{U}) \, dm \qquad (B-11)$$

Putting Eq. (B-12) into Eq. (B-11) and simplifying

$$\frac{\partial \dot{W}}{\partial \dot{\xi}_i} = \frac{\partial W}{\partial \xi_i} = W_{,i} \qquad \frac{\partial \dot{W}^T}{\partial \dot{\xi}_i} = \frac{\partial W^T}{\partial \xi_i} = W_{,i}^T \qquad (B-12)$$

$$\frac{\partial KE}{\partial \dot{\xi}_i} = \int (\vec{r}^T W_{,i}^T \dot{W} \vec{r} + \vec{r}^T W_{,i}^T \dot{W} \phi \vec{U} + \vec{r}^T \dot{W}^T W_{,i} \phi \vec{U}$$

$$+ \vec{r}^T W_{,i}^T W \phi \vec{U} + \vec{U}^T \phi^T W_{,i}^T \dot{W} \phi \vec{U} + \vec{U}^T \phi^T W_{,i}^T W \phi \vec{U}) \, dm \qquad (B-13)$$

$$\frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{\xi}_i}\right) = \int \Big(\vec{r}^{\,T} \dot{W}_{,3}^{T} \dot{W} \vec{r} + \vec{r}^{\,T} W_{,i}^{T} \ddot{W} \vec{r} + \vec{r}^{\,T} \dot{W}_{,3}^{T} \dot{W} \phi \dot{\vec{U}}$$

$$+ \vec{r}^{\,T} W_{,i}^{T} \ddot{W} \phi \vec{U} + \vec{r}^{\,T} W_{,i}^{T} \dot{W} \phi \dot{\vec{U}} + \vec{r}^{\,T} \ddot{W}^{T} W_{,i} \phi \vec{U}$$

$$+ \vec{r}^{\,T} \dot{W}^{T} \dot{W}_{,3} \phi \vec{U} + \vec{r}^{\,T} \dot{W}^{T} W_{,i} \phi \dot{\vec{U}} + \vec{r}^{\,T} \dot{W}_{,3}^{T} W \phi \dot{\vec{U}}$$

$$+ \vec{r}^{\,T} W_{,i}^{T} \dot{W} \phi \dot{\vec{U}} + \vec{r}^{\,T} W_{,i}^{T} W \phi \ddot{\vec{U}} + \dot{\vec{U}}^{T} \phi^{T} W_{,i}^{T} \dot{W} \phi \vec{U}$$

$$+ \vec{U}^{T} \phi^{T} \dot{W}_{,3}^{T} \dot{W} \phi \vec{U} + \vec{U}^{T} \phi^{T} W_{,i}^{T} \ddot{W} \phi \vec{U} + \vec{U}^{T} \phi^{T} W_{,i}^{T} \dot{W} \phi \dot{\vec{U}}$$

$$+ \dot{\vec{U}}^{T} \phi^{T} W_{,i}^{T} W \phi \dot{\vec{U}} + \vec{U}^{T} \phi^{T} \dot{W}_{,3}^{T} W \phi \dot{\vec{U}} + \vec{U}^{T} \phi^{T} W_{,i}^{T} \dot{W} \phi \dot{\vec{U}}$$

$$+ \vec{U}^{T} \phi^{T} W_{,i}^{T} W \phi \ddot{\vec{U}} \Big)\, dm \qquad (B-14)$$

Finally we express the derivative of the kinetic energy with respect to the large motion position as ξ_i.

$$\frac{\partial KE}{\partial \xi_i} = \int \Big(\vec{r}^{\,T} \dot{W}_{,3}^{T} \dot{W} \vec{r} + \vec{r}^{\,T} \dot{W}_{,3}^{T} \dot{W} \phi \vec{U} + \vec{r}^{\,T} \dot{W}^{T} \dot{W}_{,3} \phi \vec{U}$$

$$+ \vec{r}^{\,T} \dot{W}_{,3}^{T} W \phi \dot{\vec{U}} + \vec{r}^{\,T} \dot{W}^{T} W_{,i} \phi \dot{\vec{U}} + \vec{U}^{T} \phi^{T} \dot{W}_{,3}^{T} \dot{W} \phi \vec{U}$$

$$+ \vec{U}^{T} \phi^{T} \dot{W}_{,3}^{T} W \phi \dot{\vec{U}} + \vec{U}^{T} \phi^{T} \dot{W}^{T} W_{,i} \phi \dot{\vec{U}} + \dot{\vec{U}}^{T} \phi^{T} W^{T} W_{,i} \phi \dot{\vec{U}} \Big)\, dm \qquad (B-15)$$

$$\frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{\xi}_i}\right) - \frac{\partial KE}{\partial \xi_i} = \int\Big(\vec{r}^T W_{,i}^T \ddot{W} \vec{r} + \vec{r}^T W_{,i}^T \ddot{W} \phi \vec{U} + 2\vec{r}^T W_{,i}^T \dot{W} \phi \dot{\vec{U}}$$

$$+ \vec{r}^T \ddot{W}^T W_{,i} \phi \vec{U} + \vec{r}^T W_{,i}^T W \phi \ddot{\vec{U}} + \vec{U}^T \phi^T W_{,i}^T \ddot{W} \phi \vec{U}$$

$$+ 2\dot{\vec{U}}^T \phi^T \dot{W}_{,i}^T \dot{W} \phi \vec{U} + \vec{U}^T \phi^T W_{,i}^T W \phi \ddot{\vec{U}} \Big)\, dm \qquad (B-16)$$

Completing our development, the expressions for the potential energy are presented organized similarly to the previous material,

$$\frac{\partial PE}{\partial \xi_i} = -\int (\vec{r} + \phi \vec{U})^T W_{,i}^T \vec{g}\, dm = -\int \vec{r}^T W_{,i}^T \vec{g}\, dm - \int \vec{U}^T \phi^T W_{,i}^T \vec{g}\, dm \qquad (B-17)$$

The small motion equations are handled in a similar way, and are easy to derive than large motion.

$$\frac{\partial KE}{\partial \dot{\vec{U}}^T} = \int (\phi^T W^T \dot{W} \vec{r} + \phi^T W^T \dot{W} \phi \vec{U} + \phi^T W^T W \phi \dot{\vec{U}}) dm \qquad (B-18)$$

$$\frac{\partial KE}{\partial \vec{U}^T} = \int (\phi^T \dot{W}^T \dot{W} \vec{r} + \phi^T \dot{W}^T \dot{W} \phi \vec{U} + \phi^T \dot{W}^T W \phi \dot{\vec{U}}) dm \qquad (B-19)$$

$$\frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{\vec{U}}^T}\right) = \int\Big(\phi^T \dot{W}^T \dot{W} \vec{r} + \phi^T W^T \ddot{W} \vec{r} + \phi^T \dot{W}^T \dot{W} \phi \vec{U} + \phi^T W^T \ddot{W} \phi \vec{U}$$

$$+ 2\phi^T W^T \dot{W} \phi \dot{\vec{U}} + \phi^T \dot{W}^T W \phi \dot{\vec{U}} + \phi^T W^T W \phi \ddot{\vec{U}} \Big)\, dm \qquad (B-20)$$

$$\frac{d}{dt}\left(\frac{\partial KE}{\partial \dot{\vec{U}}^T}\right) - \frac{\partial KE}{\partial \vec{U}^T} = \int\Big(\phi^T W^T \ddot{W} \vec{r} + \phi^T W^T \ddot{W} \phi \vec{U}$$

$$+ 2\phi^T W^T \dot{W} \phi \dot{\vec{U}} + \phi^T W^T W \phi \ddot{\vec{U}}\,)\,dm \qquad (B-21)$$

Potential energy,

$$\frac{\partial PE}{\partial \vec{U}^T} = \int \Gamma^T C \Gamma \vec{U} dx - \int \phi^T W^T \vec{g}\, dm \qquad (B-22)$$

We can simplify these expressions further by the substution of the second time derivative of the coordinate transformation matrix W. The contributions to the coriolis and centripetal forces represented by $\ddot{W}_r$, termed the residual accelerations. The contribution to the general forces is represented by $W_{,i}$.

$$\ddot{W} = \ddot{W}_r + \sum_{i=1}^{3} W_{,i}\ddot{\xi}_i \qquad (B-23)$$

For large motion,

$$\int (\vec{r}^T W_{,i}^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r)\vec{r} + \vec{r}_r W_{,i}^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r)\phi \vec{U} + 2\vec{r}^T W_{,i}^T \dot{W} \phi \dot{\vec{U}}$$

$$+ \vec{r}^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r) W_{,i} \phi \vec{U} + \vec{r}^T W_{,i}^T W \phi \ddot{\vec{U}} + \vec{U}^T \phi^T W_{,i}^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r)\phi \vec{U}$$

$$+ 2\vec{U}^T \phi^T W_{,i}^T \dot{W} \phi \dot{\vec{U}} + \vec{U}_T \phi^T W_{,i}^T W \phi \ddot{\vec{U}} - \vec{r}^T W_{,i}^T \vec{g} - \vec{g}^T W_{,i} \phi \vec{U})dm = \vec{F}_\xi \qquad (B-24)$$

For small motion,

$$\int (\phi^T W^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r)\vec{r} + \phi^T W^T (\sum_{j=1}^{3} W_{,j}\ddot{\xi}_j + \ddot{W}_r)\phi \vec{U} + 2r\phi^T W^T \dot{W} \phi \dot{\vec{U}}$$

$$\overset{\cdot\cdot}{+}\ \phi^T W^T W \phi \vec{U} + \Gamma^T C \Gamma \vec{U} - \phi^T W^T \vec{g} = \vec{F}_u \qquad (B-25)$$

Collecting terms and arranging the coefficients the two Lagrange equations can be written as the equations of motion for the large and small generalized coordinates. It is these equations which must be solved by computer simulation code.

For the large motion,

$$\sum_{j=1}^{3} \begin{bmatrix} \int \vec{r}^T W_{,i}^T W_{,j} \vec{r}\, dm + \\ \int \vec{r}^T W_{,i}^T W_{,j} \phi \vec{U}\, dm + \\ \int \vec{U}^T \phi^T W_{,i}^T W_{,j} \phi \vec{U}\, dm + \\ \int \vec{U}^T \phi^T W_{,i}^T W_{,j} \vec{r}\, dm \end{bmatrix} \ddot{\xi}_j + \begin{bmatrix} \int \vec{r}^T W_{,i}^T W \phi\, dm + \\ \int \vec{U}^T \phi^T W_{,i}^T W \phi\, dm \end{bmatrix} \ddot{\vec{U}} =$$

$$\begin{bmatrix} F_{\xi i} - \int \vec{r}^T W_{,i}^T \ddot{W}_{,r} \vec{r}\, dm - \int \vec{r}^T W_{,i}^T \ddot{W}_{,r} \phi \vec{U}\, dm - 2\int \vec{r}^T W_{,i}^T \dot{W} \phi \vec{U}\, dm \\ - \int \vec{U}^T \phi^T W_{,i}^T \ddot{W}_{,r} \vec{r}\, dm - \int \vec{U}^T \phi^T W_{,i}^T \ddot{W}_{,r} \phi \vec{U}\, dm - 2\int \vec{U}^T \phi^T W_{,i}^T \dot{W} \phi \vec{U}\, dm \\ + \int \vec{r}^T W_{,i}^T \vec{g}\, dm + \int \vec{g}^T W_{,i} \phi \vec{U}\, dm \end{bmatrix} \qquad (B-26)$$

For the small motion,

$$\sum_{j=1}^{3} [\int \phi^T W^T W_{,j} \vec{r}\, dm + \int \phi^T W^T W_{,j} \phi \vec{U}\, dm] \ddot{\xi}_j + [\int \phi^T W^T W \phi\, dm] \ddot{\vec{U}} +$$

$$[\int 2\phi^T W^T \dot{W} \phi\, dm] \dot{\vec{U}} + [\int \phi^T \ddot{W}_{,r} \phi\, dm + \int \Gamma^T C \Gamma\, dx] \vec{U} =$$

$$\int \phi^T W^T \vec{g}\, dm - \int \phi^T W^T \ddot{W}_{,r} \vec{r}\, dm + \vec{F}_u \qquad (B-27)$$

By defining

$$M_{qq}(i,j) = \int \vec{r}^T W_{,i}^T W_{,j} \vec{r}\, dm + \int \vec{r}^T W_{,i}^T W_{,j} \phi \vec{U} dm + \int \vec{U}^T \phi^T W_{,i}^T W_{,j} \phi \vec{U} dm$$

$$+ \int \vec{U}^T \phi^T W_{,i}^T W_{,j} \vec{r}\, dm \qquad (i = 1,2,3) \qquad (j = 1,2,3) \qquad (B-28)$$

$$M_{qn} = \begin{bmatrix} \int \vec{r}^T W_{,1}^T W \phi dm + \int \vec{U}^T \phi^T W_{,1}^T W \phi dm \\ \int \vec{r}^T W_{,2}^T W \phi dm + \int \vec{U}^T \phi^T W_{,2}^T W \phi dm \\ \int \vec{r}^T W_{,3}^T W \phi dm + \int \vec{U}^T \phi^T W_{,3}^T W \phi dm \end{bmatrix} \qquad (B-29)$$

$$M_{nq} = M_{qn}^T \qquad (B-30)$$

$$M_{nn} = \int \phi^T W^T W \phi dm \qquad (B-31)$$

$$G_n = \int 2 \phi^T W^T \dot{W} \phi dm \qquad (B-32)$$

$$K_n = \int \phi^T \ddot{W}_r \phi dm + \int \Gamma^T C \Gamma dx \qquad (B-33)$$

$$F_{qi} = F_{\xi i} - \int \vec{r}^T W_{,i}^T \ddot{W}_r \vec{r}\, dm - \int \vec{r}^T W_{,i}^T \ddot{W}_r \phi \vec{U} dm - 2 \int \vec{r}^T W_{,i}^T \dot{W} \phi \vec{U} dm$$

$$- \int \vec{U}^T \phi^T W_{,i}^T \ddot{W}_r \vec{r}\, dm - \int \vec{U}^T \phi^T W_{,i}^T \ddot{W}_r \phi \vec{U} dm - 2 \int \vec{U}^T \phi^T W_{,i}^T \dot{W} \phi \vec{U} dm$$

$$+ \int \vec{r}^T W_{,i}^T \vec{g}\, dm + \int \vec{g}^T W_{,i} \phi \vec{U} dm \qquad (i = 1,2,3) \qquad (B-34)$$

$$\vec{F}_n = \int \phi^T W^T \vec{g}\, dm - \int \phi^T W^T \ddot{W}_r \vec{r}\, dm + \vec{F}_u \qquad (B-35)$$

$$\vec{F}_q = \left[F_{q1} \; F_{q2} \; F_{q3} \right]^T \qquad (B-36)$$

The final equations of motion are written as

$$M_{qq} \ddot{\vec{\xi}} + M_{qn} \ddot{\vec{U}} = \vec{F}_q \tag{B-37}$$

$$M_{nq} \ddot{\vec{\xi}} + M_{nn} \ddot{\vec{U}} + G_n \dot{\vec{U}} + K_n \vec{U} = \vec{F}_n \tag{B-38}$$

APPENDIX C. DERIVATION OF THE GENERALIZED FORCES OF THE EQUATIONS OF THE MOTION

To derive the generalized forces the principle of virtual work is used. We first write the transformation between global coordinates and local coordinates, Eqs. (C-1) and (C-2) respectively,

$$\begin{bmatrix} 1 \\ X \\ Y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ X_0 & \cos(\theta) & \sin(\theta) \\ Y_0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \qquad (C-1)$$

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -X_0\cos(\theta) - Y_0\sin(\theta) & \cos(\theta) & \sin(\theta) \\ X_0\sin(\theta) - Y_0\cos(\theta) & -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ X \\ Y \end{bmatrix} \qquad (C-2)$$

From Eqs. (C-1) and (C-2) we can find x and y.

$$x = -X_0\cos(\theta) - Y_0\sin(\theta) + X\cos(\theta) + Y\sin(\theta) \qquad (C-3)$$

$$y = X_0\sin(\theta) - Y_0\sin(\theta) + X\sin(\theta) + Y\cos(\theta) \qquad (C-4)$$

Taking derivative of the Eq. (C-3) and (C-4) doing necessary substations, we can find,

$$\delta x = \delta X\cos(\theta) + \delta Y\sin(\theta) \qquad (C-5)$$

$$\delta y = -\delta X\sin(\theta) + \delta Y\cos(\theta) \qquad (C-6)$$

Using the virtual work principle,

$$\delta W_k = [\,-T\cos(\delta + \Phi(0))v(0)](\delta\theta + \delta\Phi(0))$$

$$+ [T\cos(\delta + \Phi(0))]\delta x + [T\sin(\delta + \Phi(0))](\delta y + \delta v(0)) \qquad (C-7)$$

Based on small angle assumptions,

$$\cos(\delta + \Phi(0)) = 1 \tag{C-8}$$

and

$$\sin(\delta + \Phi(0)) = \delta + \Phi(0) \tag{C-9}$$

Doing the necessary operations and substitutions,

$$\begin{aligned}\delta W_k = {} & (T\cos(\theta) - T\delta\sin(\theta) - T\Phi(0)\sin(\theta))\delta X + (T\sin(\theta) + T\delta\cos(\theta) + \\ & T\Phi(0)\cos(\theta))\delta Y + (-Tv(0))\delta(\theta) + (T\delta + T\Phi(0))\delta v(0) \\ & + (-Tv(0)\delta\Phi(0))\end{aligned} \tag{C-9}$$

From Eq. (C-10), one can write large motion generalized forces ($\vec{F}_\xi$), i.e., Eq. (C-11) and small motion generalized forces ($\vec{F}_u$), i.e., equation (C-12),

$$\vec{F}_\xi = \begin{bmatrix} T\cos(\theta) - T\delta\sin(\theta) - T\Phi(0)\sin(\theta) \\ T\sin(\theta) + T\delta\cos(\theta) + T\Phi(0)\cos(\theta) \\ -Tv(0) \end{bmatrix} \tag{C-11}$$

$$\vec{F}_u = \begin{bmatrix} T\delta + T\Phi(0) \\ -Tv(0) \end{bmatrix} \tag{C-12}$$

APPENDIX D. COMPUTER SIMULATION CODE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%% DEFINITIONS OF VARIABLES %%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%  a0-7    =The coefficients for the numerical integration                  %
%  area    =Cross sectional area of the missile (m**2)                      %
%  beta1   =Modal node 1                                                    %
%  beta2   =Modal node 2                                                    %
%  count   =Constant that defines the element positions in vectors.         %
%  dtheta  =Desired pitch angle(dgrs)                                       %
%  dthetadot=Desired angular velocity(rd/s)                                 %
%  dtheta2dot=Desired angular acceleration(rd/s**2)                         %
%  delta   =Control angle(dgrs)                                             %
%  E       =Young's modules(pascal)                                         %
%  epsilon  =Parameter for integration coefficient                          %
%  fn      =Small motion force coefficient matrix                           %
%  flexible=Constant that does the flexible part on and off                 %
%  fnn     =Reformed small motion force coefficient matrix                  %
%  fq      =Large motion force vector                                       %
%  ftime   =Finish time of the integration                                  %
%  force   =External force(Newton)                                          %
%  gn      =Gyroscopic coefficient matrix                                   %
%  grav    =Gravitational constant(m/s**2)                                  %
%  gravvec =Gravity matrix                                                  %
%  h       =Time step used in the integration                               %
%  izz     =Moment of inertia about the z-axis(m**4)                        %
%  iota    =Parameter for integration coefficients                          %
%  kn      =Stifness matrix                                                 %
%  k       =Number of the column of the matrix                              %
%  kp,kv   =The position and velocity feedback gains                        %
%  l       =Number of the row of the matrix                                 %
%  lforce  =Large motion force vector                                       %
%  lm      =The length of the missile (meter)                               %
%  lmn     =Constant(lm*n)                                                  %
%  mn      =Reformed small motion inertia and coupling coefficient matrix   %
%  mqn     =Coupling inertia coefficient matrix                             %
%  mqq     =Large motion inertia coefficient matrix                         %
%  ma      =Constant(mu*area)                                               %
%  mphi    =Shape function matrix                                           %
%  mphi2   =Second time derivative the shape function                       %
%  mu      =Mass density of the missile(kg/m**3)                            %
%  n       =Number of the integration                                       %
%  phi     =Small motion slope position                                     %
%  phidot  =Small motion slope velocity                                     %
%  phi2dot =Small motion slope acceleration                                 %
%  rigid   =Constant that does the controller on and off                    %
%  rlocal  =Local position vector                                           %
% q        =Large motion generalized position vector                          %
% qdot     =Large motion generalized velocity vector                          %
% q2dot    =Large motion generalized acceleration vector                      %
% stif     =Stifness matrix                                                   %
% sforce   =Small motion force vector                                         %
% slope      =Trajectory path slope                                           %
% tempfirst=Temporary matrix for the Simpson's rule                           %
% templast=Temporary matrix for the Simpson's rule                            %
% templ...                                                                    %
% temp6    =Temporary matrices for integration                                %
% temp7...                                                                    %
% temp10   =Temporary matrices for large and small motion coefficients        %
% theta    =Large motion angular position (pitch angle-dgrs)                  %
% thetadot=Large motion angular velocity(rd/s)                                %
% theta2dot=Large motion angular acceleration(rd/s**2)                        %
% time     =Sec                                                               %
% u        =Small motion generalized position vector                          %
% udot     =Small motion generalized velocity vector                          %
% u2dot    =Small motion generalized acceleration vector                      %
% um       =Small motion position vector                                      %
% umdot    =Small motion velocity vector                                      %
% v        =Small motion deflection position(m)                               %
% vdot     =Small motion deflection velocity(m/s)                             %
% v2dot    =Small motion deflection acceleration(m/s**2)                      %
% wresid   =Residual acceleration matrix                                      %
% wdot     =Time derivative of the transformation matrix                      %
% wksi     =The derivative of the transformation matrix with respect to the   %
%            large motion generalized coordinates                             %
% X        =Large motion x-direction position(m)                              %
% x        =Local x coordinate                                                %
% xdot     =Large motion x-direction velocity(m/s)                            %
% x2dot    =Large motion x-direction acceleration(m/s**2)                     %
% y        =Large motion y-direction position(m)                              %
% ydot     =Large motion y-direction velocity(m/s)                            %
% y2dot    =Large motion y-direction acceleration(m/s**2)                     %
% mnn1,kn1,                                                                   %
% kn2,fn1,                                                                    %
% fn2,fn33,                                                                   %
% gn1,mqq1,                                                                   %
% fq1,fq2 =Temporary matrices for the calculation of the coefficients         %
% c1,c2,f1,                                                                   %
% f2,f3,f4=Shape function constants                                           %
%  m22,f22,                                                                   %
% h2,b2,A,                                                                    %
% B,F,temp14,                                                                 %
% temp20  =Temporary matrices for the controller                              %
% m11,m12,                                                                    %
% m21,m22,                                                                    %
% f11,h1,b1=Constants for the controller                                      %
% tplot    =Time plot vector                                                  %
% xplot    =Large motion x-direction position plot vector             %
% yplot    =Large motion y-direction position plot vector             %
% thetaplot=Large motion angular position plot vector                 %
% xdotplot =Large motion x-direction velocity plot vector             %
% ydotplot =Large motion y-direction velocity plot vector             %
% thetadotplot=Large motion angular velocity plot vector              %
% vplot    =Small motion deflection position plot vector              %
% phiplot  =Small motion slope position plot vector                   %
% vdotplot =Small motion deflection velocity plot vector              %
% phidotplot=Small motion slope velocity plot vector                  %
% Deltaplot =Control angle plot vector                                %
% dthetaplot=Desired trajectory angular position plot vector          %
% dthetadeg,deltadeg,                                                 %
% thetadeg  =Angles converted from radian to degrees                  %
%                                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% MAIN PROGRAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                LEVEL 1                                  %
%  This is the controlling program for flexible missile. It defines the   %
% variables, determines the coefficients for explicit integration and mode %
% shape function, calls large and small motion coefficient matrices routines,%
% large and small motion integration routines, controller routines, output %
% routines                                                                %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[X,y,theta,xdot,ydot,thetadot,lm,beta1,beta2,E,izz,force,...
phi,delta,v,iota,epsilon,ftime,h,grav,ma,vdot,phidot,n,lmn,flexible,..
count,mphi,mphi2,stif,gravvec,rlocal,x2dot,y2dot,theta2dot,v2dot,...
phi2dot] = constants1;

[m22,f22,h2,b2,A,F,B,dtheta2dot,dthetadot,dtheta,kv,...
kp,rigid,temp14,slope,temp20]=constants1a;

[q,qdot,q2dot,temp6,wresid,wdot,wksi,w,...
lforce,sforce,um,umdot,fnn,gn,kn,mn,count1,m11,m12,m21,f11,h1,...
b1]=constants2a(X,y,theta,xdot,ydot,thetadot,x2dot,y2dot,theta2dot);

[u,udot,u2dot,mnn1,kn1,kn2,fn1,fn2,fn33,mnn,gn1,mqq,mqn,...
fq,mqq1,temp1,temp2,temp3,temp4,temp5,temp7,mqn1,temp8,fq1,temp9,...
temp10,fq2] = constants2b(v,phi,vdot,phidot,v2dot,phi2dot);

[tplot,xplot,yplot,...
thetaplot,xdotplot,ydotplot,thetadotplot,vplot,phiplot,vdotplot,...
phidotplot,thetadeg,deltaplot,dthetaplot,dthetadeg,deltadeg] =...
constants3(ftime,h);

[a0,a1,a2,a3,a4,a5,a6,a7] = ccoef(epsilon,iota,h);
```

```
[c1,c2,f1,f2,f3,f4]=mphicons(lm,beta1,beta2);

for time = 0.0:h:ftime,
   [w,wdot,wksi,wresid,lforce,sforce,um,umdot] = wmatderivvariant(...
   theta,X,y,xdot,ydot,thetadot,force,phi,delta,v,vdot,phidot,...
   wresid,wdot,wksi,w,lforce,sforce,um,umdot);

   [mqq,mqn,fq,fq2]=lrgcof(wksi,rlocal,mphi,um,w,lforce,wresid,...
   wdot,umdot,gravvec,ma,lm,n,mqq,mqn,fq,mqq1,temp7,mqn1,...
   temp8,fq1,temp9,temp10,lmn,c1,c2,f1,f2,f3,f4,beta1,beta2,fq2);
   if flexible==1,

   [mn,gn,kn,fnn]=smlcof(mphi,w,mqn,mqq,fq,wdot,wresid,mphi2,stif,...
   gravvec,sforce,ma,lm,n,lmn,fnn,gn,kn,mn,mnn1,kn1,kn2,fn1,fn2,fn33,...
   mnn,rlocal,c1,c2,f1,f2,f3,f4,beta1,beta2);

   [u,udot,u2dot]=intsml(a0,a1,a2,a3,a4,a5,a6,a7,u,udot,u2dot,...
   temp1,temp2,temp3,temp4,temp5,mn,gn,fnn,kn);

   end


   [q,qdot,q2dot] = intlrg(h,a0,a3,a6,a7,mqn,mqq,fq,q,qdot,q2dot,...
temp6,u2dot);

   [X,y,theta,xdot,ydot,thetadot,v,phi,vdot,phidot,x2dot,...
   y2dot,theta2dot,v2dot,phi2dot]=chvar(q,qdot,q2dot,u,udot,u2dot);

   if rigid==1,
     [delta,dtheta,temp20]=rigidcontrol(mqq,fq2,force,theta,m11,m12,m21,m22,...
     f11,f22,h1,h2,b1,b2,A,F,B,dtheta2dot,thetadot,dthetadot,...
     dtheta,kv,kp,temp14,time,slope,temp20);
   end

   [tplot,xplot,yplot,thetaplot,xdotplot,count,deltaplot,dthetaplot] =gplot(...
   count,time,X,y,theta,xdot,tplot,xplot,yplot,thetaplot,xdotplot,thetadeg,...
   delta,deltaplot,dtheta,dthetadeg,dthetaplot,deltadeg);

   [ydotplot,thetadotplot,vplot,phiplot,vdotplot,phidotplot,count]=gplot2(...
   count,ydot,thetadot,v,phi,vdot,phidot,ydotplot,thetadotplot,vplot,phiplot,..
   vdotplot,phidotplot);

   count1=count1+1

   if count1==0,
     force=0;
   end

   if count1==0,
     t1=X;
     t2=y;
```

```
    t3=theta;
    t4=xdot;
    t5=ydot;
    t6=thetadot;
    t7=v;
    t8=phi;
    t9=vdot;
    t10=phidot;
    t11=x2dot;
    t12=y2dot;
    t13=theta2dot;
    t14=v2dot;
    t15=phi2dot;
    t16=delta;

    clear X y theta xdot ydot thetadot v phi vdot phidot delta
    clear fn1 fn2 fn33 fnn fq fq1 gn gn1 kn kn1 kn2 lforce mn mnn
    clear mnn1 mqn mqn1 mqq mqq1 q qdot q2dot sforce temp1 temp2
    clear temp3 temp4 temp5 temp6 temp7 temp8 temp9 u udot u2dot
    clear um umdot w wdot wksi wresid count1 temp10
    clear x2dot y2dot theta2dot v2dot phi2dot
    clear m11 m12 m21 f1 h1 b1 temp11 temp12 temp13 tempfirst templast

    X=t1;
    y=t2;
    theta=t3;
    xdot=t4;
    ydot=t5;
    thetadot=t6;
    v=t7;
    phi=t8;
    vdot=t9;
    phidot=t10;
    x2dot=t11;
    y2dot=t12;
    theta2dot=t13;
    v2dot=t14;
    phi2dot=t15;
    delta=t16;

    [q,qdot,q2dot,temp6,wresid,wdot,wksi,w,...
    lforce,sforce,um,umdot,fnn,gn,kn,mn,count1,m11,m12,m21,f1,h1,...
    b1]=constants2a(X,y,theta,xdot,ydot,thetadot,x2dot,y2dot,...
    theta2dot);

    [u,udot,u2dot,mnn1,kn1,kn2,fn1,fn2,fn33,mnn,gn1,mqq,mqn,...
    fq,mqq1,temp1,temp2,temp3,temp4,temp5,temp7,mqn1,temp8,fq1,temp9,...
    temp10] = constants2b(v,phi,vdot,phidot,v2dot,phi2dot);
    end
end
```

```
[plot1,plot2,plot3,plot4,plot5,plot6]=allplot1(tplot,xplot,yplot,...
thetaplot,xdotplot,ydotplot,thetadotplot,dthetaplot)
[plot7,plot8,plot9,plot10,plot11]=allplot2(tplot,vplot,phiplot,...
vdotplot,phidotplot,deltaplot)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  LEVEL 2                                 %
%   This function determines constant values and initial values of the     %
% variables and matrices.                                                  %
%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [X,y,theta,xdot,ydot,thetadot,lm,beta1,beta2,E,izz,force,...
phi,delta,v,iota,epsilon,ftime,h,grav,ma,vdot,phidot,n,lmn,flexible,..
count,mphi,mphi2,stif,gravvec,rlocal,x2dot,y2dot,theta2dot,v2dot,...
phi2dot] = constants1

area=0.0113097;
lm=4;
beta1=4.712398/lm;
beta2=7.853981/lm;
count=0;
delta=0;
E=2.0e+11;
epsilon=0.25;
flexible=1;
force=30000;
ftime=0.4;
grav=-9.8066;
h=0.001;
iota=0.5;
izz=0.000010178;
mu=7861.05;
n=10;
lmn=lm/n;
ma=area*mu;
phi=0;
phidot=0;
theta=pi/4;
thetadot=0;
X=0;
xdot=0;
v=0;
vdot=0;
ydot=0;
y=0;
mphi=zeros(3,2);
mphi2=zeros(3,2);
stif=zeros(3);
```

```
 stif(3,3)=E*izz;
 gravvec=zeros(3,1);
 gravvec(3)=grav;
 rlocal=zeros(3,1);
 x2dot=0;
 y2dot=0;
 theta2dot=0;
 v2dot=0;
 phi2dot=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%   This function determines constant values and initial values of the      %
% variables and matrices.                                                    %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function[m22,f22,h2,b2,A,F,B,dtheta2dot,dthetadot,dtheta,kv,...
kp,rigid,temp14,slope,temp20]=constants1a

m22=0;
f22=0;
h2=0;
b2=0;
A=0;
F=0;
B=0;
dtheta2dot=0;
dthetadot=0;
dtheta=0.7854;%45 Degree
kv=6;
kp=9;
rigid=1;
temp14=zeros(2);
slope=-2.6180;
temp20=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%   This function determines constant values and initial values of the      %
% variables and matrices.                                                    %
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [q,qdot,q2dot,temp6,wresid,wdot,wksi,w,...
lforce,sforce,um,umdot,fnn,gn,kn,mn,count1,m11,m12,m21,f11,h1,...
b1]=constants2a(X,y,theta,xdot,ydot,thetadot,x2dot,y2dot,theta2dot)
```

```
q=zeros(3,1);
qdot=zeros(3,1);
q2dot=zeros(3,1);
temp6=zeros(3,1);
wresid=zeros(3);
wdot=zeros(3);
wksi=zeros(3,9);
w=zeros(3);
lforce=zeros(3,1);
sforce=zeros(2,1);
um=zeros(2,1);
umdot=zeros(2,1);
fnn=zeros(2,1);
gn=zeros(2);
kn=zeros(2);
mn=zeros(2);
count1=0;
q(1)=x;
q(2)=y;
q(3)=theta;
qdot(1)=xdot;
qdot(2)=ydot;
qdot(3)=thetadot;
q2dot(1)=x2dot;
q2dot(2)=y2dot;
q2dot(3)=theta2dot;
m11=zeros(2);
m12=zeros(2,1);
m21=zeros(1,2);
f11=zeros(2,1);
h1=zeros(2,1);
b1=zeros(2,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function determines constant values and initial values of the       %
% variables and matrices.                                                     %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [u,udot,u2dot,mnn1,kn1,kn2,fn1,fn2,fn33,mnn,gn1,mqq,mqn,...
fq,mqq1,temp1,temp2,temp3,temp4,temp5,temp7,mqn1,temp8,fq1,temp9,...
temp10,fq2] = constants2b(v,phi,vdot,phidot,v2dot,phi2dot)

u=zeros(2,1);
udot=zeros(2,1);
u2dot=zeros(2,1);
mnn1=zeros(2);
kn1=zeros(2);
kn2=zeros(2);
```

```
fn1=zeros(2,1);
fn2=zeros(2,1);
fn33=zeros(2,1);
mnn=zeros(2);
gn1=zeros(2);
mqq=zeros(3);
mqn=zeros(3,2);
fq=zeros(3,1);
mqq1=zeros(3);
temp1=zeros(2,1);
temp2=zeros(2,1);
temp3=zeros(2,1);
temp4=zeros(2,1);
temp5=zeros(2,1);
temp7=zeros(3);
mqn1=zeros(3,2);
temp8=zeros(3);
fq1=zeros(3,1);
fq2=zeros(3,1);
temp9=zeros(3);
temp10=zeros(3);
u(1)=v;
u(2)=phi;
udot(1)=vdot;
udot(2)=phidot;
u2dot(1)=v2dot;
u2dot(2)=phi2dot;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                         %
%   This function determines the initial values of the output matrices.   %
%                                                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [tplot,xplot,yplot,...
thetaplot,xdotplot,ydotplot,thetadotplot,vplot,phiplot,vdotplot,...
phidotplot,thetadeg,deltaplot,dthetaplot,dthetadeg,deltadeg] = ...
constants3(ftime,h)


tplot=zeros(1,(ftime/h)+1);
xplot=zeros(1,(ftime/h)+1);
yplot=zeros(1,(ftime/h)+1);
thetaplot=zeros(1,(ftime/h)+1);
xdotplot=zeros(1,(ftime/h)+1);
ydotplot=zeros(1,(ftime/h)+1);
thetadotplot=zeros(1,(ftime/h)+1);
vplot=zeros(1,(ftime/h)+1);
phiplot=zeros(1,(ftime/h)+1);
```

```
vdotplot=zeros(1,(ftime/h)+1);
phidotplot=zeros(1,(ftime/h)+1);
thetadeg=0;
deltaplot=zeros(1,(ftime/h)+1);
dthetaplot=zeros(1,(ftime/h)+1);
dthetadeg=0;
deltadeg=0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%   Formulating the coefficients used in the sequential integration method. %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [a0,a1,a2,a3,a4,a5,a6,a7] = ccoef(epsilon,iota,h)
a0=1/(epsilon*(h*h));
a1=iota/(epsilon*h);
a2=1/(epsilon*h);
a3=0.5/epsilon-1;
a4=iota/epsilon-1;
a5=h/2*(a4-1);
a6=h*(1-iota);
a7=iota*h;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%   Formulating the coefficients used in the mode shape function matrix.    %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [c1,c2,f1,f2,f3,f4]=mphicons(lm,beta1,beta2)

c1=(sin(beta1*lm)-sinh(beta1*lm))/(-cos(beta1*lm)+cosh(beta1*lm));
c2=(sin(beta2*lm)-sinh(beta2*lm))/(-cos(beta2*lm)+cosh(beta2*lm));
e=2*beta2-2*(c2/c1)*beta1;
f1=1/(2*c1)+(c2*beta1)/((c1^2)*e);
f2=-(c2/(c1*e));
f3=-(beta1/(c1*e));
f4=1/e;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                           %
%   This function calculates the matrices which changes with time.          %
%                                                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [w,wdot,wksi,wresid,lforce,sforce,um,umdot]=...
wmatderivvariant(theta,X,y,xdot,ydot,thetadot,force,phi,delta,v,...
vdot,phidot,wresid,wdot,wksi,w,lforce,sforce,um,umdot)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the transformation matrix w                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
w(1,1)=1;
w(2,1)=X;
w(2,2)=cos(theta);
w(2,3)=-sin(theta);
w(3,1)=y;
w(3,2)=sin(theta);
w(3,3)=cos(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the time derivative of the transformation matrix  WDOT    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wdot(2,1)=xdot;
wdot(2,2)=-thetadot*sin(theta);
wdot(2,3)=-thetadot*cos(theta);
wdot(3,1)=ydot;
wdot(3,2)=thetadot*cos(theta);
wdot(3,3)=-thetadot*sin(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Computation of the derivative of the transformation matrix with respect  %
% to X, Y, theta: WKSI                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wksi(2,1)=1;
wksi(3,4)=1;
wksi(2,8)=-sin(theta);
wksi(3,8)=cos(theta);
wksi(2,9)=-cos(theta);
wksi(3,9)=-sin(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Computation of the the residual acceleration matrix WRESID                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
wresid(2,2)=-(thetadot^2)*cos(theta);
wresid(2,3)=(thetadot^2)*sin(theta);
wresid(3,2)=-(thetadot^2)*sin(theta);
wresid(3,3)=-(thetadot^2)*cos(theta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the large motion force matrix                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lforce(1)=force*(cos(theta)-delta*sin(theta)-phi*sin(theta));
lforce(2)=force*(sin(theta)+delta*cos(theta)+phi*cos(theta));
lforce(3)=-force*v;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the small motion force matrix                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sforce(1)=force*(phi+delta);
sforce(2)=v*force;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the small motion position vector                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
um(1)=v;
um(2)=phi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Computation of the small motion velocity vector                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
umdot(1)=vdot;
umdot(2)=phidot;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                            %
%   This function calculates values for the large motion coefficient matrices%
%                                                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [mqq,mqn,fq,fq2]=lrgcof(wksi,rlocal,mphi,um,w,lforce,...
wresid,wdot,umdot,gravvec,ma,lm,n,mqq,mqn,fq,mqq1,temp7,mqn1,...
temp8,fq1,temp9,temp10,lmn,c1,c2,f1,f2,f3,f4,beta1,beta2,fq2)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Creating the coefficient matrix  for MQQ                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:3,
  temp1=wksi(:,3*(i-1)+1:3*i);
  for j=1:3,
    temp2=wksi(:,3*(j-1)+1:3*j);
    temp7=temp1'*temp2;
    mqq1(i,j)=quadmat('astring11',1,1,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,...
    f4,mphi,0,0,rlocal,temp1,temp2,um,0,0,0,0,temp7,0,0,...
    0,0,0);
  end
end
mqq=ma*(mqq1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Creating the coefficient matrix for MQN                                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:3,
    temp=wksi(:,3*(i-1)+1:3*i);
    temp8=temp'*w;
    mqn1(i,:)=quadmat('astring22',1,2,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,...
    f4,mphi,w,0,rlocal,0,0,um,temp,0,0,0,0,temp8,0,0,0,0);
end
mqn=ma*(mqn1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Creating the coefficient mayrix for FQ                                   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Computation of the FQ coefficient
for i=1:3,
    temp=wksi(:,3*(i-1)+1:3*i);
    temp9=temp'*wresid;
    temp10=temp'*wdot;
    fq1(i)=quadmat('astring33',1,1,n,lm/n,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,...
    mphi,0,wresid,rlocal,0,0,um,temp,wdot,umdot,gravvec,0,0,temp9,...
    temp10,0,0);
end
fq2=ma*fq1;
fq=fq2+lforce;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %                                                                            %
 %   This function calculates values for small motion coefficient matrices.   %
 %                                                                            %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


 function [mn,gn,kn,fnn]=smlcof(mphi,w,mqn,mqq,fq,wdot,wresid,mphi2,...
 stif,gravvec,sforce,ma,lm,n,lmn,fnn,gn,kn,mn,mnn1,kn1,...
 kn2,fn1,fn2,fn33,mnn,rlocal,c1,c2,f1,f2,f3,f4,beta1,beta2)

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   Creating the coefficient matrix  MN                                      %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 mnn=quadmat('astring44',2,2,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,...
     mphi,w,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
 mnn1=ma*mnn;
 mn=mnn1-mqn'*inv(mqq)*mqn;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   Creating the coefficient matrix  GN                                      %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 gn1=quadmat('astring55',2,2,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,...
     mphi,w,0,0,0,0,0,0,wdot,0,0,0,0,0,0,0,0);
 gn=ma*gn1;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %   Creating the coefficient matrix  KN                                      %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 kn1=quadmat('astring66',2,2,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,...
    mphi,w,wresid,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
kn2=quadmat('astring77',2,2,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,...
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,mphi2,stif);
kn=ma*kn1+kn2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Creating the coefficient matrix  FN                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fn1=quadmat('astring88',2,1,n,lmn,ma,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
    wresid,rlocal,0,0,0,0,0,0,gravvec,0,0,0,0,0,0);
fn33=ma*(fn1) + sforce;
fnn=fn33-mqn'*inv(mqq)*fq;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function implicitly integrates the small motion equation.           %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [u,udot,u2dot]=intsml(a0,a1,a2,a3,a4,a5,a6,a7,u,udot,...
u2dot,temp1,temp2,temp3,temp4,temp5,mn,gn,fnn,kn)

   temp1=a0*u+a2*udot+a3*u2dot;
   temp2=a1*u+a4*udot+a5*u2dot;
   temp3=mn*temp1;
   temp4=gn*temp2;
   temp1=fnn+temp3+temp4;
   temp2=u;
   temp5=kn+a0*mn+a1*gn;
   temp1=temp5\temp1;
   u=temp1;
   temp3=u2dot;
   u2dot=a0*(u-temp2)-a2*udot-a3*temp3;
   udot=udot+a6*temp3+a7*u2dot;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function explicitly integrates the large motion equation.            %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [q,qdot,q2dot]=intlrg(h,a0,a3,a6,a7,mqn,mqq,fq,q,qdot,...
q2dot,temp6,u2dot)

q=q+h*qdot+a3*q2dot/a0;
qdot=qdot+a6*q2dot;
temp6=temp6+mqn*u2dot;
```

```
fqn=fq-temp6;
q2dot=inv(mqq)*fqn;
qdot=qdot+q2dot*a7;
q=q+q2dot/a0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function updates the time dependent values.                         %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [X,y,theta,xdot,ydot,thetadot,v,phi,vdot,phidot,x2dot,...
y2dot,theta2dot,v2dot,phi2dot]=chvar(q,qdot,q2dot,u,udot,u2dot)

X=q(1);
y=q(2);
theta=q(3);
xdot=qdot(1);
ydot=qdot(2);
thetadot=qdot(3);
x2dot=q2dot(1);
y2dot=q2dot(2);
theta2dot=q2dot(3);
v=u(1);
phi=u(2);
vdot=udot(1);
phidot=udot(2);
v2dot=u2dot(1);
phi2dot=u2dot(2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function controls rigid or flexible missile with rigid-body          %
% controller. The control angle(delta) is limited to 10 degrees.              %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function[delta,dtheta,temp20]=rigidcontrol(mqq,fq2,force,theta,m11,m12,...
m21,m22,f11,f22,h1,h2,b1,b2,A,F,B,dtheta2dot,thetadot,dthetadot,...
dtheta,kv,kp,temp14,time,slope,temp20)

if (time >= 0) & (time < 0.1),
  dtheta2dot=0;
  dthetadot=0;
  dtheta=pi/4;
  temp20=dtheta;
elseif time >= 0.1,
  dtheta2dot=0;
  dthetadot=slope;
  dtheta=temp20+slope*(time-0.1);
end

m11(1,1)=mqq(1,1);
m11(1,2)=mqq(1,2);
m11(2,1)=mqq(2,1);
m11(2,2)=mqq(2,2);
m12(1)=mqq(1,3);
m12(2)=mqq(2,3);
m21(1)=mqq(3,1);
m21(2)=mqq(3,2);
m22=mqq(3,3);



f11(1)=fq2(1);
f11(2)=fq2(2);
f22=fq2(3);

h1(1)=force*cos(theta);
h1(2)=force*sin(theta);

b1(1)=-force*sin(theta);
b1(2)=force*cos(theta);
temp14=inv(m11);
A=m22-m21*temp14*m12;
F=f22-m21*temp14*(f11+h1);
B=m21*temp14*b1;

delta=-inv(B)*(A*(dtheta2dot-kv*(thetadot-dthetadot)-kp*(theta-...
      dtheta))-F);

if delta >= 0.174532 , %10 degree
  delta=0.174532;
end
if delta <= -0.174532,
  delta=-0.174532;
elseif (delta < 0.174532) | (delta > -0.174532),
  delta=delta;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                          %
%   This function calculates the elements of matrices for output           %
%                                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function  [tplot,xplot,yplot,thetaplot,xdotplot,count,deltaplot,...
dthetaplot] =gplot(count,time,X,y,theta,xdot,tplot,xplot,yplot,...
thetaplot,xdotplot,thetadeg,delta,deltaplot,dtheta,dthetadeg,...
dthetaplot,deltadeg)

count=count+1
thetadeg=(180*theta)/pi;
dthetadeg=(180*dtheta)/pi;
deltadeg=(180*delta)/pi;
tplot(count)=time;
xplot(count)=X;
yplot(count)=y;
thetaplot(count)=thetadeg;
xdotplot(count)=xdot;
deltaplot(count)=deltadeg;
dthetaplot(count)=dthetadeg;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
%   This function calculates the elements of matrices for output         %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [ydotplot,thetadotplot,vplot,phiplot,vdotplot,phidotplot,count]=...
gplot2(count,ydot,thetadot,v,phi,vdot,phidot,ydotplot,thetadotplot,...
vplot,phiplot,vdotplot,phidotplot)

ydotplot(count)=ydot;
thetadotplot(count)=thetadot;
vplot(count)=v;
phiplot(count)=phi;
vdotplot(count)=vdot;
phidotplot(count)=phidot;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                        %
%   This function plots the output graphs.                               %
%                                                                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [plot1,plot2,plot3,plot4,plot5,plot6]=allplot1(tplot,xplot,yplot,...
thetaplot,xdotplot,ydotplot,thetadotplot,dthetaplot)

plot1=plot(tplot,xplot),pause
plot2=plot(tplot,yplot),pause
plot3=plot(tplot,thetaplot,'--',tplot,dthetaplot,'+'),pause
plot4=plot(tplot,xdotplot),pause
plot5=plot(tplot,ydotplot),pause
plot6=plot(tplot,thetadotplot),pause
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                %
%   This function plots the output graphs.                                       %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function [plot7,plot8,plot9,plot10,plot11]=allplot2(tplot,vplot,phiplot,...
vdotplot,phidotplot,deltaplot)

plot7=plot(tplot,vplot),pause
plot8=plot(tplot,phiplot),pause
plot9=plot(tplot,vdotplot),pause
plot10=plot(tplot,phidotplot),pause
plot11=plot(tplot,deltaplot),pause


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                  LEVEL 3                                       %
%   This function will integrate the coefficient matrices of small and large %
% motion over the length of the missile using Simpson's integration method.  %
%                                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function aa = quadmat(F, l, k, n, int, ma,c1,c2,beta1,beta2,...
f1,f2,f3,f4,mphi,w,wresid,rlocal,temp1,temp2,um,temp,wdot,umdot,...
gravvec,temp7,temp8,temp9,temp10,mphi2,stif)


temp11=zeros(1,k*n);
temp12=zeros(1,k);
temp13=zeros(1,k);
tempfirst=zeros(1,k);
templast=zeros(1,k);
aa=zeros(1,k);


x=0;
for i=1:n,
   x=x+int;
   temp11(:,k*(i-1)+1:i*k)=feval(F,x,c1,c2,beta1,beta2,f1,f2,f3,f4,...
   mphi,w,wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,...
   gravvec,temp7,temp8,temp9,temp10,mphi2,stif);

end

for m= 1:k,
  for i = 2:2:(n-1),
    temp12(:,m) = temp12(:,m) + temp11(:, k*(i-1)+m);
  end
end
```

```
for p= 1:k,
  for j = 3:2:(n-1),
    temp13(:,p) = temp13(:,p) + temp11(:, k*(j-1)+p);
  end
end

tempfirst(:,1:k)=temp11(:,1:k);
templast(:,1:k)=temp11(:,n*k-(k-1):n*k);
aa=(int/3)*(tempfirst+4*temp13+2*temp12+templast);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%   This function creates the product of matrices for integration of MQQ %
% coefficient matrix.                                                   %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y11=astring11(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

rlocal(1)=1;
rlocal(2)=x;

y11=rlocal'*temp7*rlocal+rlocal'*temp7*mphi*um+um'*mphi'*temp7*...
mphi*um+um'*mphi'*temp7*rlocal;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
%   This function creates the product of matrices for integration of MQN %
% coefficient matrix.                                                   %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y22=astring22(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

rlocal(1)=1;
rlocal(2)=x;

y22=rlocal'*temp8*mphi+um'*mphi'*temp8*mphi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%   This function creates the product of matrices for integration of FQ         %
% coefficient matrix.                                                           %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y33=astring33(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

rlocal(1)=1.0;
rlocal(2)=x;

y33=-rlocal'*temp9*rlocal-rlocal'*temp9*mphi*um-2*rlocal'*temp10*...
mphi*umdot-um'*mphi'*temp9*rlocal-um'*mphi'*temp9*mphi*um-2*...
um'*mphi'*temp10*mphi*umdot+rlocal'*temp'*gravvec+gravvec'*temp*...
mphi*um;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                               %
%   This function creates the product of matrices for integration of MN         %
% coefficient matrix.                                                           %
%                                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y44=astring44(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
```

```
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

y44=mphi'*w'*w*mphi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%  This function creates the product of matrices for integration of GN %
% coefficient matrix.                                                  %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y55=astring55(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

y55=2*mphi'*w'*wdot*mphi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%  This function creates the product of matrices for integration of KN %
% coefficient matrix.                                                  %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y66=astring66(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));
```

```
mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

y66=mphi'*w'*wresid*mphi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function creates the product of matrices for integration of KN      %
% coefficient matrix.                                                         %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y77=astring77(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi2(3,1)=f1*(beta1^2)*(c1*(-cos(beta1*x)+cosh(beta1*x))...
-sin(beta1*x)+sinh(beta1*x))+f3*(beta2^2)*(c2*...
(-cos(beta2*x)+cosh(beta2*x))-sin(beta2*x)+sinh(beta2*x));

mphi2(3,2)=f2*(beta1^2)*(c1*(-cos(beta1*x)+cosh(beta1*x))...
-sin(beta1*x)+sinh(beta1*x))+f4*(beta2^2)*(c2*...
(-cos(beta2*x)+cosh(beta2*x))-sin(beta2*x)+sinh(beta2*x));

y77=mphi2'*stif*mphi2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                             %
%   This function creates the product of matrices for integration of FN      %
% coefficient matrix.                                                         %
%                                                                             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


function y88=astring88(x,c1,c2,beta1,beta2,f1,f2,f3,f4,mphi,w,...
wresid,rlocal,temp1,temp2,um,temp,rlocal,wdot,umdot,gravvec,temp7,...
temp8,temp9,temp10,mphi2,stif)

mphi(3,1)=f1*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f3*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

mphi(3,2)=f2*(c1*(cos(beta1*x)+cosh(beta1*x))+sin(beta1*x)+...
sinh(beta1*x))+f4*(c2*(cos(beta2*x)+cosh(beta2*x))+...
sin(beta2*x)+sinh(beta2*x));

rlocal(1)=1;
rlocal(2)=x;

y88=mphi'*w'*gravvec-mphi'*w'*wresid*rlocal;
```

LIST OF REFERENCES

1. Blakelock, J. H., *Automatic Control of Aircraft and Missiles,* pp.236, John Wiley & Sons,Inc., 1965

2. Jenkins, P. N., "Missile Dynamics Equations For Guidance and Control Modeling and Analysis," Technical Report RG-84-17, April 1984

3. Greenwood, D. T., *Principles of Dynamics*, Prentice-Hall, Inc. 1988

4. Chang, L., and Gannon, K, K., "A Dynamic Model on a Single- Flexible Manipulator," *1987 Design Technology Conference - 11. Biennial Conference on Mechanical Vibrations and Noise, ASME Modal and Testing Analysis,* pp.23, DE-Vol.3, September 1987

5. Meirovitch, L., *Elements Of Vibration Analysis*, pp.220-227, McGraw-Hill, Inc. 1986

6. Chang, L., and Hamilton, J. F., "A Sequential Integration Method," ASME *Journal of Dynamic Systems, Measurement and Control*, pp.382, Vol. 110, December 1988

7. Laufenberg, R, S., *Computer Simulation of a Rotational Single-Element Flexible Spacecraft Boom,* M.S. Thesis, Naval Postgraduate School, Monterey, California, March 1987

8. Gannon, K. P., *Modeling and Experimental Validation of a Single-Link Flexible Manipulator,* M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1986

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 69Hy Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	2
4.	Professor Liang-Wey Chang,Code 69Ck Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	2
5.	Professor Morris Driels, Code 69Dr Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
6.	Professor Fotis Papoulis, Code 69Pa Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
7.	Professor Harold A.Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5000	1
8.	Mehmet Aysel Tepekoy Koyonu Cad. No:64 Pasabahce-Istanbul / TURKEY	5

9. Naval Sea Systems Command 1
Code 5760-900
Weapon and Combat Systems
National Center Bldg. 2
2521 Jefferson Davis Hwy
Arlington VA, 22202

10. Deniz Kuvvetleri Komutanligi 1
Personel Daire Baskanligi
Bakanliklar-Ankara / TURKEY

11. Deniz Harp Okulu Komutanligi 1
Tuzla - Istanbul / TURKEY

12. Ortadogu Teknik Universitesi 1
Ankara / TURKEY

13. Istanbul Teknik Universitesi 1
Makina Fakultesi, Gumussuyu
Istanbul / TURKEY

14. Bogazici Universitesi 1
P.K. 2, Bebek
Istanbul / TURKEY

614-583

Thesis
A9973 Aysel
c.1 The computer simulation and modeling of a flexible missile in 2-D motion.

Thesis
A9973 Aysel
c.1 The computer simulation and modeling of a flexible missile in 2-D motion.

AMERICAN BINDERY
2
TOPEKA KANSAS

The computer simulation and modeling of
3 2768 000 89091 7
DUDLEY KNOX LIBRARY